UNIVERSITÉ DU QUÉBEC À MONTRÉAL (TÉLÉ-UNIVERSITÉ)

ET

INSTITUT NATIONAL DES SCIENCES APPLIQUÉES DE ROUEN

GESTION DE L'ÉVOLUTION DES ONTOLOGIES : MÉTHODES ET OUTILS POUR UN RÉFÉRENCEMENT SÉMANTIQUE ÉVOLUTIF FONDÉ SUR UNE ANALYSE DES CHANGEMENTS ENTRE VERSIONS D'ONTOLOGIE

THÈSE

PRÉSENTÉE EN COTUTELLE

COMME EXIGENCE PARTIELLE

DU DOCTORAT EN INFORMATIQUE COGNTIVE

PAR

DELIA CODRUTA ROGOZAN

MAI 2008

À mes chers parents,

et à ma très jolie nièce Anne-Justine…

REMERCIEMENTS

J'aimerais remercier et exprimer ma profonde reconnaissance aux personnes qui m'ont aidée, orientée et soutenue tout au long de mon doctorat.

Pour la Télé-université, je tiens à remercier mes deux directeurs de thèse, Gilbert Paquette et Richard Hotte. Je leur suis particulièrement reconnaissante pour leur confiance, leurs encouragements répétés, leurs critiques et leurs conseils toujours opportuns, leur rigueur scientifique et notamment, pour toutes nos discussions animées autour des concepts manipulés par cette thèse. Qu'ils soient persuadés de ma plus profonde considération.

Pour l'INSA de Rouen, je tiens à remercier Habib Abdulrab, mon directeur, pour m'avoir donné la chance de commencer ce doctorat à l'Institut National des Sciences Appliquées de Rouen (France) et de le continuer, grâce à une cotutelle de thèse, à la Télé-Université (UQAM - Canada). Je lui suis reconnaissante pour son intérêt et la patience qu'il a démontrée à mon égard pendant toutes ces années.

Mes remerciements vont également aux membres du jury Roger Nkambou, Claude Rigault, Michel Gagnon et Jean-Pierre Pécuchet, pour l'évaluation de mes travaux de recherche et pour leurs suggestions pertinentes ainsi qu'à l'équipe de direction du programme de doctorat en Informatique cognitive pour son support et son aide.

Je ne saurai oublier les collègues et amis du LICEF, notamment Julien Contamines, Lucie Moulet et Valéry Psyché pour leur soutien et pour la qualité des discussions enrichissantes autour des problématiques abordées dans ma thèse et également Mihai Tabaras, pour son aide à résoudre mes problèmes techniques. Je désire mentionner aussi l'appui chaleureux de Michel Léonard, Karin Lundgren-Cayrol, Diane Ruelland, Olga Marino, Marcelo Maïna, Claire Banville, Ioan Rosca. Ils ont tous su, à leur manière, me seconder durant ces années de thèse, parfois éreintantes, mais toujours enrichissantes.

Je termine en soulignant ma reconnaissance à mes parents, Ioan et Iuliana, et à ma sœur Alexandrina. Ils ont su m'encourager jour après jour et ont toujours cru en moi. Sans leur soutien constant et affectif, je n'aurais jamais pu avancer sur le chemin ardu du doctorat. Et enfin, je remercie Gabriel, pour sa présence et sa grande patience, sans qui je ne serais peut-être pas arrivée au bout de cette aventure.

TABLE DES MATIERES

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ACRONYMES

CHB : *ChangeHistoryBuilder*

OC : *OntologyChange*

SAM : *SemanticAnnotationModifier*

UKI : *UniformKnowledgeIdentifier*

LORIT : Laboratoire-Observatoire de Recherche en Ingénierie du Téléapprentissage

LORNET : *Learning Object Repository Network*

OWL : *Web Ontology Language*

RDF : *Resource Description Framework*

URI : *UniformResourceIdentifier*

XML *: eXtensible Markup Language*

RÉSUMÉ

Dans cette thèse, nous abordons les problématiques liées à la gestion de l'évolution des ontologies, dans le cadre défini par le Web sémantique. Nous le faisons en développant l'idée qu'une « bonne évolution » est celle qui, en plus de préserver la consistance de l'ontologie évoluée, préserve aussi l'intégrité de ce qui repose déjà sur l'ontologie. Dans notre contexte, ceci est le référencement sémantique des ressources. Ce dernier aspect nécessite toutefois une connaissance approfondie des changements apportés à l'ontologie. Sans cette connaissance, il nous serait impossible d'identifier les effets de l'évolution sur le référencement sémantique et donc, de maintenir l'accès et l'interprétation des ressources au moyen de l'ontologie évoluée.

À l'heure actuelle, très peu de travaux de recherche proposent des solutions pour la journalisation des changements (et leur identification après l'évolution d'ontologie). Encore moins nombreux sont les travaux qui analysent les effets de l'évolution sur les ressources référencées et aucun, à notre connaissance, ne propose des solutions concrètes pour la résolution des effets problématiques, conduisant à une perte d'accès aux ressources, par exemple.

Dans cette thèse, nous apportons des solutions novatrices pour combler ces lacunes. Nous proposons une approche de journalisation des changements, élémentaires et complexes, apportés à une version d'ontologie V_N pour obtenir une nouvelle version V_{N+1}. Cette approche est fondée sur un modèle uniformisé, mais aussi riche sémantiquement, car il a comme base une ontologie des changements, également développée dans cette thèse. Nous fournissons aussi un formalisme de représentation des changements, semi-compatible avec l'OWL, qui rend interopérable et partageable l'historique des changements. De même, nous contribuons de façon originale à l'avancement de la problématique du Web sémantique en proposant un ensemble de stratégies de modification du référencement sémantique affecté par l'évolution des ontologies. Ces stratégies sont fondées sur l'analyse des types de changement et de leur impact sur l'accès aux ressources référencées, mais aussi sur l'interprétation de celles-ci au moyen de la nouvelle version de l'ontologie.

Nous regroupons toutes ces solutions dans un cadre, intégrateur et modulaire, composé du système CHB (*ChangeHistoryBuilder*), pour la gestion de l'historique des changements, mais aussi du système SAM (*SemanticAnnotationModifier*), pour la gestion du référencement sémantique des ressources. L'originalité de ce cadre s'impose par le fait que, loin d'être dépourvu de toute intervention humaine, il s'incarne dans une société distribuée d'agents humains et logiciels qui s'échangent des informations et des services pendant, mais surtout après l'évolution des versions d'ontologie. Soulignons aussi qu'il s'est précisé pendant notre avancement dans la conception d'une méthodologie d'évolution des ontologies. Ceci fait de notre cadre une réponse pertinente aux besoins technologiques de notre méthodologie, également proposée dans cette thèse.

Mots-clés : Web sémantique, évolution des ontologies, méthodologie, journalisation des changements, référencement sémantique évolutif.

INTRODUCTION

Web sémantique, référencement des ressources, ontologies et évolution des ontologies sont les quatre notions au cœur de cette thèse. Ces notions, rassemblées dans une structure significative, pourront ouvrir « la connaissance et les travaux humains à l'analyse signifiante d'agents logiciels donnant accès à une nouvelle classe d'outils grâce auxquels nous pourrons vivre, travailler, apprendre ensemble », comme l'affirment Sir Berners-Lee et ses collaborateurs dans un désormais célèbre article fondateur, *The Semantic Web* (Revue *Scientific American*, 2001).

Dans cette introduction, nous présentons d'abord notre contexte scientifique suivi de quelques éléments de problématique. Puis, nous énonçons nos objectifs de recherche qui débouchent sur une anticipation des résultats de nos travaux. Nous terminons par l'esquisse de la démarche de recherche mise en œuvre et par un plan qui indique les lignes directrices de cette thèse.

Contexte scientifique

Web sémantique et référencement des ressources

La vision du **web sémantique**, proposée premièrement par Berners-Lee, Hendler, et Lasilla (2001), fait d'abord référence à un vaste espace d'échange des ressources entre les humains et les agents logiciels permettant une meilleure exploitation des informations et des services variés. Espace virtuel de demain, il verra alors les humains déchargés d'une partie de leurs tâches de recherche d'informations et de combinaison des résultats grâce aux capacités accrues des agents logiciels à accéder aux contenus de ressources et à effectuer des

raisonnements sur ceux-ci. Pour cela, les ressources doivent être sémantiquement référencées à l'aide des descripteurs interprétables informatiquement.

Dans le cadre du web sémantique, plusieurs termes sont utilisés pour dénommer ces descripteurs : *métadonnées*, *annotations* (Bechhofer et Goble, 2001; Prie et Garlatti, 2004) ou encore *références sémantiques* (Paquette et Rosca, 2004). La différenciation entre les uns et les autres est presque insaisissable, la plupart des scientifiques les utilisant pour dénoter la description, structurée, explicite et formelle, du contenu d'une ressource à partir des éléments standards ou des connaissances disponibles dans une ou plusieurs ontologies. De plus, comme indiqué par Laublet, Reynaud et Charlet, (2002), le terme 'sémantique' désigne la volonté de faire émerger le sens du contenu d'une ressource dans sa description/référencement et ce, afin d'améliorer le processus de recherche d'informations, sa compréhension, et sa réutilisation par les humains, mais aussi par les agents logiciels du web.

Pour uniformiser la terminologie, nous utilisons dans cette thèse l'expression **référencement sémantique** pour désigner la représentation formelle du contenu d'une ressource. Ce référencement se compose d'une ou plusieurs **références sémantiques**, associées ou intégrées aux ressources, ces références étant principalement des identifiants des concepts (ou classes) appartenant aux diverses ontologies formelles.

Ontologies et évolution

Nous observons que la représentation explicite du contenu des ressources est rendue disponible grâce aux ontologies. Depuis les années 90, les ontologies sont au cœur des travaux en ingénierie des connaissances et, depuis la 'révolution' du web sémantique, elles se retrouvent parmi les couches basiques de l'architecture du futur web (Berners-Lee, 2000). La raison de cette popularité est, en partie, due au fait que les ontologies, en proposant une compréhension commune et partagée d'un domaine, autant par les humains que par les agents logiciels (Borst, 1997; Gruninger et Lee, 2002), permettent de résoudre des problèmes d'interprétation et d'interopérabilité (Wache et *al.*, 2001).

Les ontologies sont définies dans plusieurs articles (Chandrasekaran, Josephson, et Benjamins, 1999; Gomez-Perez, 1999; Gruber, 1993; Guarino, 1997). Brièvement, une **ontologie** est une collection ordonnée des descriptions de classes, de propriétés et

d'instances. Une classe décrit un concept du domaine spécifié par l'ontologie. Une instance est un individu concret appartenant à cette classe. Une propriété est une relation qui énonce un type d'interaction entre les classes. Une ontologie est complètement définie quand les classes sont structurées par leurs propriétés et que ces propriétés sont combinées dans des axiomes permettant d'effectuer des inférences sur le contenu de l'ontologie.

Les ontologies peuvent être représentées selon différents langages (Fensel, Harmelen, Horrocks, McGuinness, et Patel-Schneider, 2001; Heflin, 2001; Oberle, Volz, Motik, et Staab, 2004). Dans le contexte du web sémantique, la plupart des ontologies sont cependant représentées à l'aide du **langage standard OWL** (*Web Ontology Language*)[1], qui s'appuie sur le modèle et le schéma RDF (*Ressource Description Framework*)[2] mais qui ajoute des constructeurs langagiers provenant principalement de la logique des descriptions.

Finalement, les ontologies doivent évoluer pour répondre aux : (a) modifications du domaine conceptualisé ; (b) besoins changeants des divers acteurs humains ou logiciels ; (c) points de vues différents sur la conceptualisation spécifiée par les ontologies (Charlet, Bachimont, et Troncy, 2003; Ding, Fensel, Klein, Omelayenko, et Schulten, 2004; Euzenat et all., 2001; Heflin et Hendler, 2000; Klein et Fensel, 2001; McGuinness, 2000; Noy et Musen, 2003a; Stojanovic et Motik, 2002; Studer, 2003). Si les travaux sur les ontologies, leur construction et leur utilisation, datent d'une dizaine d'années (OntoWeb, 2002a), ceux sur **l'évolution de l'ontologie** sont bien plus récents et leur poursuite est indispensable pour que les ontologies servent toute application du web sémantique.

Éléments de problématique

Le problème qui nous préoccupe concerne **la gestion de l'Févolution des ontologies du web sémantique** ainsi que **la préservation de l'intégrité[3] du référencement sémantique** qui y repose. Particulièrement, dans cette thèse nous essayons de répondre à

---

[1] La description du langage OWL peut être consultée à : http://www.w3.org/TR/owl-features/.
[2] La description du RDF peut être consultée à l'adresse : http://www.w3.org/TR/2004/REC-rdf-primer-20040210/.
[3] Par la notion de **préservation de l'intégrité du référencement sémantique**, nous comprenons le maintien de l'accès aux ressources et une interprétation correcte de ces dernières, au moyen de la nouvelle version de l'ontologie, issue de l'évolution.

certains questionnements soulevés par l'évolution des ontologies, questionnements que nous énonçons ci-après.

1.) L'évolution des ontologies nécessite-t-elle un support méthodologique ; lequel serait le plus approprié ?

Comme tout processus complexe, l'évolution des ontologies nécessite un cadre méthodologique qui la structure. Actuellement, plusieurs approches pour la construction des ontologies sont proposées dans la documentation scientifique du domaine, mais très peu qui traitent de l'évolution sur le plan méthodologique (Haase et Sure, 2004). De plus, aucun consensus ne se dégage à propos de ces dernières approches, vraisemblablement à cause d'une perspective différente sur ce qu'est l'évolution : dans un contexte centralisé, elle signifie l'application des changements et l'utilisation de l'ontologie évoluée (Maedche, Motik, et Stojanovic, 2003; Stojanovic, 2004) ; dans un contexte décentralisé, elle est considérée comme l'identification et la gestion des versions multiples pour une même ontologie (Klein, 2004; Klein et Noy, 2003).

Par conséquent, un de nos intérêts dans cette thèse est d'analyser les deux perspectives d'évolution, d'extraire les étapes nécessaires à la gestion de l'évolution des ontologies et finalement, d'organiser ces étapes sous la forme d'un processus cohérent.

2.) Quels sont les types et les degrés de complexité des changements qu'on peut apporter à une ontologie ?

Pour exprimer l'évolution, nous avons besoin de recenser l'ensemble des modifications que l'on peut apporter aux ontologies. Ce vocabulaire est essentiellement un ensemble des changements typés, chaque changement ayant une signification bien précise. Les changements peuvent être élémentaires, comme l'ajout ou l'effacement, mais aussi plus complexes, comme le déplacement ou la fusion, ces derniers ayant l'avantage d'une sémantique plus riche, nous permettant d'exprimer des modifications de plus haut niveau.

Dans cette thèse, nous envisageons d'étudier les types de changements, aussi élémentaires que complexes, qui peuvent s'effectuer sur une ontologie, de les caractériser et les classifier, mais aussi de trouver le moyen de les représenter formellement afin de

permettre aux agents logiciels d'interpréter leur signification. Ce dernier aspect est requis par le contexte des ontologies formelles du web sémantique, dans lequel nous situons notre recherche.

3.) Que signifie la gestion des changements apportés aux ontologies ?

La gestion des changements apportés à une ontologie doit se réaliser en deux temps : pendant et après l'évolution d'une ontologie. Pendant l'évolution, elle concerne l'édition des changements et l'utilisation des stratégies pour préserver la consistance de l'ontologie. C'est ce qu'on appelle la **phase d'évolution**. Plusieurs travaux de recherche, dont certains proposent même des outils de construction et de modification des ontologies (OntoWeb, 2002b), couvrent une bonne partie des exigences de cette phase d'évolution, à l'exception de l'édition des changements complexes et de la journalisation[4], structurée et standardisée, des processus d'évolution des ontologies.

Après l'évolution, cette gestion concerne l'identification des changements et la résolution des effets des changements sur le référencement des ressources : c'est ce qu'on appelle la phase de **mise en opération de l'évolution**. Cette phase est indispensable dans le contexte du web sémantique (et de tout système fondé sur le référencement sémantique des ressources), sachant que les effets des changements peuvent conduire à une perte de l'accès aux ressources référencées ou même à une interprétation inconsistante de celles-ci.

4.) Après l'évolution, comment peut-on identifier les changements apportés à l'ontologie d'une manière riche sémantiquement ?

Deux approches sont proposées dans la documentation actuelle du domaine : (a) l'approche par journalisation (Maedche, Motik, et Stojanovic, 2003; Oliver, Shahar, Musen, et Shortliffe, 1999; Stojanovic, 2004), fondée sur l'identification des changements à partir d'une trace d'évolution, qui est habituellement un fichier-journal où sont enregistrées les opérations effectuées lors de la phase d'évolution ; (b) l'approche par alignement, fondée sur la comparaison des versions d'ontologie au niveau structurel et sur l'utilisation d'un

[4] Par journalisation, nous comprenons l'enregistrement, dans un fichier-journal, des opérations effectuées par les utilisateurs lors de la modification de l'ontologie.

algorithme pour identifier des modifications (Klein, Fensel, Kiryakov, et Ognyanov, 2002; Noy et Musen, 2002). Dans cette thèse, nous avons choisi l'approche par journalisation : elle est la seule à pouvoir récupérer, d'une manière intégrale et non ambiguë, l'ensemble des changements, élémentaires ou complexes, apportés à l'ontologie et nous donner ainsi la possibilité de connaître l'effet de chaque changement sur le référencement sémantique des ressources.

De plus, tout en choisissant l'approche par journalisation pour l'identification des changements, nous cherchons dans cette thèse un moyen d'éviter les problèmes d'accès et d'interprétation des fichiers-journal qui sont souvent stockés localement et formalisés dans un langage trop spécifique à l'outil. Nous cherchons également un modèle qui nous permettrait de journaliser des changements ayant divers degrés de complexité

5.) Comment peut-on garantir une 'bonne évolution', sachant qu'il faudrait maintenir l'intégrité du référencement sémantique qui repose sur l'ontologie ?

L'évolution de l'ontologie peut causer une perte d'accès aux ressources référencées. Prenons comme exemple une ressource R, référencée par la classe Professeur, et un changement qui fusionne `Professeur` et `DesignerPedagogique`. Après l'évolution, la ressource R n'est plus disponible pour une requête telle que « donnez-moi toutes les personnes ayant le titre de `Professeur` ». Nous disons alors que l'accès à cette ressource est brisé.

Peu nombreuses sont les recherches actuelles qui identifient les effets potentiels de l'évolution sur les ressources référencées (Heflin et Hendler, 2000; Oliver, Shahar, Musen, et Shortliffe, 1999; Stuckenschmidt et Klein, 2003b) et cela, uniquement pour quelques changements élémentaires, qui ajoutent ou effacent des classes ou des propriétés. Encore moins important est le nombre de recherches qui abordent la modification du référencement pour éviter l'effet problématique de certains changements. À notre connaissance, seul le modèle CREAM (Handschuh, Staab, et Maedche, 2001) propose des recommandations à cet effet, sans offrir cependant une solution concrète.

Dans cette thèse, nous envisageons de combler ces lacunes en identifiant les effets que les changements peuvent avoir sur le référencement sémantique, mais aussi en cherchant un moyen pour résoudre les effets les plus problématiques.

Objectifs de recherche

Nos travaux de recherche visent à répondre aux questionnements formulés. Actuellement, aucune recherche ne donne une réponse complète à l'ensemble de ces questions, comme nous le faisons dans cette thèse.

Nos travaux concernent la conception d'un support à la gestion des changements apportés aux ontologies du web sémantique, qui sont des ontologies formelles, représentées à l'aide du langage standard OWL. Ce support devrait se situer autant au niveau conceptuel, en proposant de nouvelles idées, notions et solutions, qu'au niveau technologique, en proposant de nouvelles technologies 'exploratoires'. D'une manière plus spécifique, nos objectifs de recherche s'énoncent comme suit :

1. Méthodologie d'évolution d'ontologies (objectif sur le plan conceptuel). Notre premier objectif de recherche est de concevoir une méthodologie présentant une vue unifiée sur l'évolution, son déroulement, ses exigences et ses besoins, les principes mis en jeu et les règles à respecter lors de l'évolution de l'ontologie dans un environnement dynamique et distribué sur le web sémantique.

2. Support à la gestion de l'historique des changements (objectif sur le plan conceptuel et technologique). La gestion de l'historique des changements est un aspect primordial de la mise en opération de l'évolution. En effet, sans connaître les changements apportés aux ontologies, on ne peut comprendre ni la signification de l'évolution, ni ses effets sur le référencement sémantique des ressources. Notre deuxième objectif vise à : (a) décrire les changements, autant élémentaires que complexes, qui peuvent être apportés aux ontologies OWL-DL[5] et les classifier sous

[5] Le langage OWL-DL, fondé sur la logique de descriptions, est une version décidable d'OWL. Ce langage possède une expressivité maximale sans perte de calculabilité, c'est-à-dire que toutes les inférences sont calculables et le calcul se réalise en un temps fini.

une forme taxonomique; (b) concevoir une méthode et un outil pour tracer l'historique des changements d'une manière qui préserve toute la richesse de l'évolution, c'est-à-dire qui journalise des changements de haut niveau de complexité (p.ex. la fusion, le déplacement, la division), mais aussi qui formalise cet historique dans un langage plus standardisé.

3. Support à la gestion du référencement sémantique (objectif sur le plan conceptuel et technologique). Notre troisième objectif est celui de trouver des solutions pour résoudre les effets problématiques, engendrés par l'évolution des ontologies, qui ont comme conséquence une perte d'accès aux ressources référencées. Pour cela, nous envisageons de concevoir une méthode et un outil nous permettant d'identifier les effets des changements (en termes d'accès et d'interprétation de ressources référencées) et d'en tenir compte pour rétablir l'accès aux ressources par une modification du référencement sémantique.

Contributions apportées

Pour répondre aux objectifs de recherche, nous avons obtenu les résultats suivants :

Pour l'objectif 1

1. Une définition de ce qu'est l'évolution des ontologies, qui unifie les perspectives existant actuellement dans un cadre cohérent et plus complet.

2. Une ébauche d'une méthodologie d'évolution des ontologies dans le contexte du web sémantique, qui identifie les étapes essentielles à l'évolution et qui décrit le déroulement prévu pour chacune de ces étapes.

Par rapport aux travaux actuels, l'apport principal de nos deux résultats est celui d'unifier les vues sur ce qu'est l'évolution et d'offrir des repères méthodologiques essentiels à l'évolution des ontologies, mais aussi à la mise en opération de l'ontologie évoluée.

Pour l'objectif 2

3. Une ontologie des changements qui est composée d'une hiérarchie des changements élémentaires et d'une hiérarchie des changements complexes ainsi que d'un certain nombre de propriétés et d'axiomes qui explicitent la signification de chaque changement. Les changements spécifiés doivent correspondre à ceux que l'on peut effectuer sur des ontologies OWL-DL.

Le fait de concevoir une ontologie de changements est très novateur dans le domaine, très peu de taxonomies étant proposées actuellement. L'apport principal de notre ontologie est celui de décrire explicitement les types des changements élémentaires et complexes, d'une manière qui, à notre connaissance, est la plus complète à l'heure actuelle.

4. Un modèle de métadonnées, fondé sur l'ontologie des changements, permettant aux divers éditeurs de journaliser des changements, élémentaires et complexes, d'une manière uniformisée, cohérente et complète.

5. Un langage, nommé *OntologyChange* (OC), proposant un nombre de constructeurs langagiers qui, combinés à ceux offerts par le langage OWL, doivent permettre la formalisation de tout type de changement et de ses conséquences.

6. Un système, nommé *ChangeHistoryBuilder* (CHB), ayant deux fonctionnalités de base : (a) journaliser les changements lors de l'évolution en utilisant le modèle de métadonnées proposé, de formaliser ensuite les changements à l'aide du langage OC+OWL, et d'intégrer à la nouvelle ontologie la trace ainsi formalisée ; (b) récupérer cette nouvelle ontologie, à la demande soit des utilisateurs soit des divers agents logiciels, d'interpréter la trace des changements et de fournir en sortie l'ensemble des changements apportés.

Le modèle de métadonnées ainsi que le langage OC, les deux à la base du système CHB, sont eux aussi des contributions novatrices dans le domaine de la gestion des changements apportés aux ontologies. Aucune recherche ne propose actuellement un langage standardisé pour la formalisation des changements, ni un modèle de journalisation qui soit adaptable aux divers éditeurs d'ontologies. Le CHB offre également un moyen d'intégrer la trace des changements formalisés à l'intérieur de la

nouvelle ontologie, ce qui nous permet d'éviter les problèmes d'accès aux fichiers-journal.

Pour l'objectif 3

7. Une analyse des effets des changements, autant élémentaires que complexes, qui s'articule autour de trois lignes directrices : (a) l'effet[6] sur l'accès direct et indirect aux ressources ; (b) l'effet sur l'interprétation des ressources ; (c) l'effet sur les relations logiques entre les classes appartenant à l'ancienne et à la nouvelle ontologie, en termes d'équivalence, d'inclusion, etc.

8. Un ensemble de stratégies pour la modification du référencement sémantique affecté par des changements problématiques, comme l'effacement des classes, la fusion ou la division, par exemple.

9. Un système, nommé *SemanticAnnotationModifier* (SAM), offrant trois services aux utilisateurs. Le premier est une mise en correspondance entre le référencement des ressources et les changements apportés à l'ontologie, afin d'identifier les changements problématiques. Le deuxième est un service d'information aux utilisateurs, leur présentant l'analyse des effets des changements problématiques. Le troisième service concerne la modification du référencement[7] affecté selon un mode (semi)automatique ou assisté.

Comme nous l'avons déjà précisé, très peu de recherches traitent de l'effet des changements sur l'accès aux ressources référencées et aucune ne propose une solution pour résoudre les effets problématiques. Dans ce contexte, les trois résultats que nous proposons pour SAM sont essentiels à une 'bonne évolution' des ontologies, mais aussi

[6] L'effet des changements sur l'accès aux ressources (direct ou indirect) est plus problématique que les deux autres types d'effets. C'est pourquoi, dans un premier temps, nous avons orienté la conception du support à la gestion du référencement sémantique (*cf.* objectif 3) en fonction de cet effet.

[7] Dans le contexte actuel de SAM, le référencement prend la forme d'une liste des URI spéciaux. Un URI, c'est-à-dire un *UniformRessourceIdentifier* (Berners-Lee, 2005), est une courte chaîne de caractères qui identifie explicitement une ressource physique ou abstraite. La syntaxe de cette chaîne respecte une norme d'Internet mise en place pour le World Wide Web.

très innovateurs. Ils offrent de nouvelles fonctions pour l'exploration des effets des changements et pour la modification du référencement sémantique des ressources.

Démarche de recherche

Du besoin particulier des environnements d'apprentissage (par projet) en ligne, vers une problématique plus large dans le cadre du web sémantique, qui débouche sur des questions propres à la gestion des changements apportés aux ontologies. Cette phrase traduit bien la démarche de notre recherche.

Tout d'abord, notre recherche portait sur la construction d'un espace d'apprentissage (par projet) en ligne, dont les composantes (acteurs, activités et objets d'apprentissage) sont référencées sémantique à l'aide des ontologies du domaine et de la tâche (Rogozan, Hotte, et Abdulrab, 2002). Cependant, les ontologies, vues comme des structures immuables, ne peuvent rendre compte, ni de l'évolution des connaissances d'apprenants, ni du caractère, essentiellement constructif et flexible, de l'apprentissage par projet. Par conséquent, il est indispensable d'offrir aux acteurs la possibilité d'adapter, pendant leur démarche de projet, les ontologies utilisées.

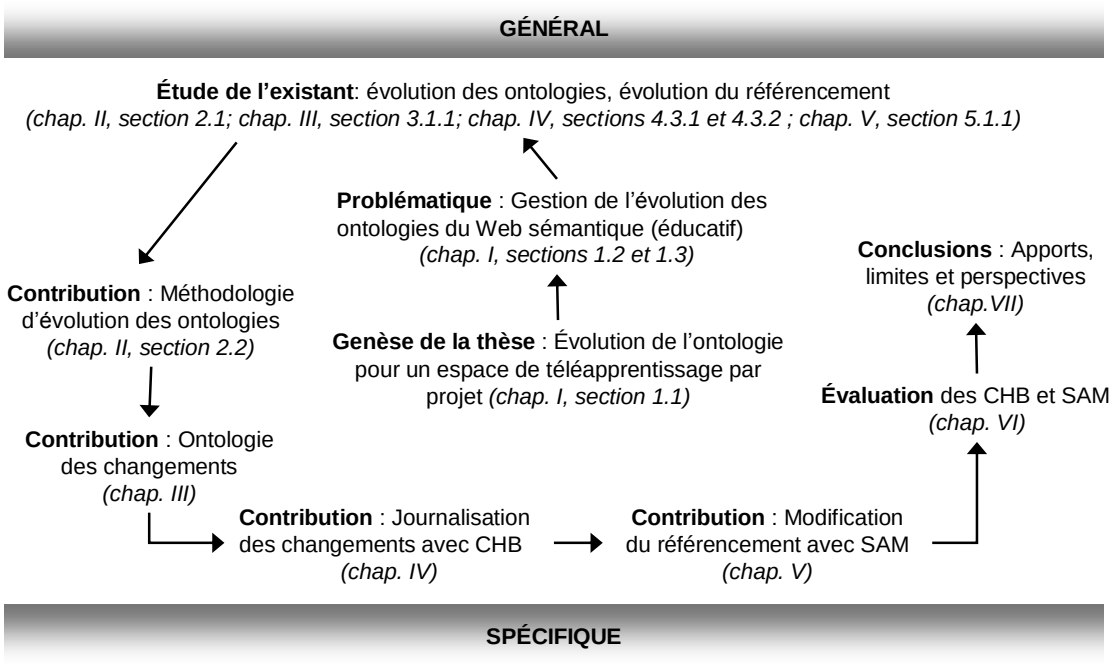
Nous sommes ainsi arrivée à définir la problématique de recherche qui est au cœur de cette thèse : l'évolution des ontologies et la mise à jour du référencement sémantique, dans le contexte du web sémantique (éducatif). En analysant ensuite l'évolution des ontologies, au sens large, nous avons identifié des besoins spécifiques, tels que l'identification (ou la journalisation) des changements, l'analyse des effets de changements et la modification du référencement des ressources, des besoins auxquels nous répondons dans cette thèse en proposant deux systèmes : le CHB et le SAM.

Pour concevoir ces deux systèmes, nous avons adopté un certain nombre de principes : (1) développer des 'prototypes exploratoires' des idées, non des systèmes finalisés ; (2) assurer une conception modulaire, permettant une utilisation de CHB et de SAM, de manière séparée ou en combinaison; (3) considérer, et non pas éviter l'intervention humaine dans le cadre des services offerts par SAM. Pour évaluer les deux systèmes, nous avons choisi le critère d'utilité, qui traite de la pertinence d'un système. Ce choix est motivé par le fait qu'on se situe dans un contexte novateur, où ce qui compte est l'apport de nouvelles idées et ce

qu'on doit mesurer est l'intérêt suscité par les systèmes qu'implémentent ces idées, les buts qu'ils permettront d'atteindre ainsi que le degré d'atteinte de ces buts.

Plan de la thèse

La figure ci-dessous montre l'organisation de la thèse par chapitres et par sections.



Organisation de la thèse par chapitres et par sections

Le **chapitre I** présente une synthèse de nos travaux antérieurs qui montrent la nécessité de faire évoluer les ontologies à la base des systèmes d'apprentissage par projet, en ligne. Il expose également le contexte de recherche, la problématique et les objectifs de cette thèse.

Le **chapitre II** est consacré à l'évolution des ontologies et se divise en trois parties. La première partie explore les méthodes actuelles de gestion de l'évolution des ontologies. La deuxième présente l'ébauche de notre méthodologie, conçue pour intégrer les avantages des méthodes présentées en première partie. La troisième partie contient une analyse des outils de construction des ontologies, analyse qui nous a permis d'identifier les besoins technologiques ressentis pour supporter la méthodologie d'évolution.

Le **chapitre III** décrit la conceptualisation et la formalisation de l'ontologie des changements que nous avons développée pour expliciter les changements qui peuvent être apportés à des ontologies OWL_DL.

Le **chapitre IV** porte sur le système CHB. Il présente le modèle de métadonnées pour la journalisation, le langage OC utilisé pour la formalisation des changements ainsi que les fonctionnalités du CHB pour récupérer, formaliser et archiver les changements, de même que pour identifier les changements, après l'évolution.

Le **chapitre V** porte sur le système SAM. Dans une première partie, il décrit la notion du référencement sémantique qui sera utilisée dans cette thèse. Dans une deuxième partie, il présente l'analyse des effets des changements sur le référencement des ressources. Finalement, la troisième partie illustre la modification du référencement affecté par les changements selon un mode (semi)automatique ou assisté.

Le **chapitre VI** décrit le protocole et la mise en œuvre de l'évaluation des systèmes CHB et SAM. Ce chapitre discute également les résultats issus de l'analyse de l'évaluation des systèmes.

Le **chapitre VII** présente les conclusions de la thèse. Il fait un retour sur la problématique et les objectifs de recherche et montre l'originalité de nos contributions. Il décrit les apports de la thèse, mais aussi ses limites et ses perspectives.

Chapitre I

ÉLÉMENTS DE PROBLÉMATIQUE

Nous présentons d'abord nos travaux antérieurs (*cf.* Section 1.1) qui montrent la nécessité de faire évoluer les ontologies à la base des environnements d'apprentissage (par projet) en ligne. Nous exposons ensuite notre contexte de recherche (*cf.* Section 1.2) qui privilégie le Web sémantique comme infrastructure pour supporter des systèmes d'apprentissage en ligne, fondés sur le référencement sémantique de ressources. Nous terminons ce chapitre en rappelant notre problématique, à savoir comment supporter l'évolution des ontologies du web sémantique, les objectifs et l'organisation de la recherche (*cf.* Section 1.3).

1.1 **Genèse de la thèse**

Cette section présente brièvement le cheminement qui nous a amenés vers le questionnement fondamental de notre recherche. Ainsi, dans un premier temps, nous nous sommes intéressée à la modélisation d'un Espace technologique de support à la Réalisation des Projets d'Apprentissage à Distance (ERPAD) par des étudiants universitaires en génie[8]. Nous avons travaillé sur la façon de concilier l'apprentissage par projet, qui est typiquement une démarche active où les apprenants sont les principaux artisans de leurs projets, avec la formation en ligne qui est fondée sur une conception préalable de cours et sur l'utilisation, quasi exclusive, des technologies de l'information et de la communication (TIC). Dans la section 1.1.1. nous présentons la notion d'apprentissage par projet et dans la section 0nous présentons le modèle de l'ERPAD dont l'armature est constituée d'ontologies appelées à évoluer dans le temps.

1.1.1. **Apprentissage par projet : caractéristiques et mise en ligne**

L'apprentissage par projet, dont l'origine remonte à Dewey (1922), et à Kilpatrick (1918), est un modèle éducatif qui transforme les apprenants en artisans de leur propre formation en organisant l'apprentissage autour de projets ouverts. Les projets sont définis comme des tâches complexes et multidisciplinaires, inspirées de la vie réelle, qui : (1) engagent les apprenants dans des activités authentiques de conception, d'investigation, de réflexion et de prise de décision ; (2) donnent aux apprenants la possibilité de travailler d'une manière autonome, sur des périodes de temps définies; (3) demandent une facilitation de la part d'un formateur et non un guidage serré ; (4) possèdent des buts précis, à la fois définis et accomplis par les apprenants, qui sont les seuls responsables de la démarche de projet ; (5) nécessitent des objectifs éducationnels explicites et non une planification prédéfinie des activités d'apprentissage; (6) mettent en place des situations propices à la coopération et à la collaboration entre apprenants en vue de permettre la conception collective des solutions; (7) s'achèvent par des biens livrables sous la forme de produits, documents ou présentations

[8] Des étudiants de l'Institut National des Sciences Appliquées (INSA de Rouen, France),

(Diehl, Grobe, Lopez, et Cabral, 1999; Lebrun, 2002; Moursund, 1999; Thomas, 2000). L'apprentissage par projet permet ainsi aux apprenants de se former à un mode d'organisation du travail qui correspond au virage vers la nouvelle société du savoir, société qui réclame une population non seulement plus instruite, plus autonome et plus créative, mais aussi mieux préparée à participer à des activités collectives de capitalisation de connaissances et de gestion de projets.

Bien qu'il possède un nombre de caractéristiques favorables à la mise en ligne, par exemple l'autonomie des apprenants ou l'utilisation des ressources informationnelles, ce modèle pose néanmoins un certain nombre de difficultés (Fung, 1996; Kim et Lee, 2002; Oliver, Shahar, Musen, et Shortliffe, 1999; Thomas, 2000) dont nous en parlons brièvement dans le Tableau I-1.

Tableau I-1 Problèmes de la mise en place d'un téléapprentissage par projet

Dimension	Problème(s) rencontré par les apprenants	Piste(s) de solution
Dynamique du projet	- Guider la démarche du projet afin de permettre, en même temps, la réalisation d'un projet personnel et la construction des connaissances demandées par le curriculum. - Organiser et gérer, d'une manière réaliste, la réalisation du projet. - Auto-évaluer et réviser continûment les connaissances et les produits du projet.	- Rendre disponible l'armature du modèle des connaissances visées et permettre aux apprenants de l'enrichir et l'adapter en fonction de leur projet. - Rendre disponible des modèles de *workflow* de projet que les apprenants peuvent ensuite réutiliser et modifier en fonction de leur besoins dans le projet. - Offrir la possibilité de confrontation des travaux partiels entre les apprenants.
Collaboration/ Interaction	- Échanger des idées et des résultats de projet. - Assurer des rétroactions bénéfiques à l'apprentissage par projet.	- Créer des situations propices à l'émergence des interactions ciblées sur les activités. - Intégrer les activités individuelles dans un réseau formé des apprenants en interaction.
Accès à l'information	- Rechercher et identifier les ressources adéquates à leur projet. - Recevoir des informations sur les aides à obtenir pendant la démarche de projet.	- Faciliter une « recherche intelligente » des ressources référencées sémantiquement. - Fournir un accès continu à des informations ciblées sur les travaux des apprenants.

1.1.2. **Espace d'apprentissage par projet à distance, fondé sur des ontologies évolutives**

En vue de résoudre ces problèmes, nous avons proposé la modélisation d'un Espace technologique de Réalisation des Projets d'Apprentissage à Distance (ERPAD) (Rogozan, 2003; Rogozan, Hotte, et Abdulrab, 2002). Cet espace, qui porte essentiellement sur la dynamique des relations inter-projets et la capitalisation de leur expérience, supporte les apprenants distants dans l'atteinte d'un certain niveau de compétence par la réalisation de divers projets d'apprentissage. Pour répondre aux problèmes de collaboration/interaction et d'accès à l'information, ERPAD intègre un Espace Dynamique de Collaboration (EDC) et un Espace Dynamique de Documentation (EDD), les deux étant construits par les apprenants à partir d'ontologies évolutives.

1.1.2.1. **Apprenants comme concepteurs de leur ERPAD**

L'apprenant comme concepteur est une dimension essentielle de l'apprentissage par projet. La justification de cette affirmation réside dans la nature de ce type d'apprentissage : une démarche constructive où les apprenants sont maîtres de leur apprentissage, de leur quête de solutions, une démarche située dans un contexte socioculturel spécifique qui détermine la forme et le contenu du projet, une démarche flexible et indépendante où les apprenants réalisent le projet sans que leurs activités soient au préalable prédéfinies ou imposées.

Vus dans cette perspective, l'EDC et l'EDD doivent être construits par les apprenants d'une manière émergente, à l'image d'un chemin se définissant pendant que nous le parcourons (Rogozan, Paquette, et Hotte, 2003). Dans ce contexte, bien plus qu'un environnement d'apprentissage qui s'adapte au parcours de l'apprenant en fonction de ses résultats, l'apprentissage par projet nécessite un environnement qui est, en partie, défini par les utilisateurs et qui, à son tour, oriente leurs activités. Nous empruntons à Bourguin et Derycke, (2005), le terme *co-évolutif* pour caractériser l'ERPAD. Ce terme traduit tant les ajustements apportés par les utilisateurs au comportement du système pour qu'il réponde à leurs besoins émergents, que les ajustements que l'ERPAD peut apporter aux apprenants en les orientant dans leur démarche de projet.

1.1.2.2. **Construire l'EDC et l'EDD avec les ontologies**

Selon Leontiev, (1984), chaque activité, dans notre cas un projet d'apprentissage, est orientée par un objet qui en motive l'existence, comme la question-problème à la base du projet. Chaque activité est produite par une somme d'actions, comme les tâches accomplies pour réaliser effectivement le projet, répondant à des buts particuliers, intermédiaires par rapport à ce qui motive l'activité. Une action peut être souvent poly-motivée, c'est-à-dire qu'une seule et même action peut réaliser diverses activités, peut passer d'une activité à une autre, manifestant ainsi sa relative indépendance.

1.1.2.2.1. *Espace de collaboration et espace de documentation*

L'espace dynamique de collaboration – l'EDC – est construit par les apprenants qui, même en travaillant sur des projets différents, peuvent interagir ponctuellement et d'une manière pertinente grâce à leurs actions communes, dirigées par des buts proches (*cf.* Figure I-1, a). Ici, l'acte d'interagir n'est pas un simple transfert d'information depuis l'expéditeur vers le destinataire, mais plutôt le modelage d'un monde commun au moyen d'actions conjuguées, une activité à part entière engageant plusieurs acteurs réalisant des activités différentes, mais interconnectées. Nous obtenons ainsi un espace spécifique, mais non pas prédéfini, car il est construit par les apprenants, dans leur contexte de projet.

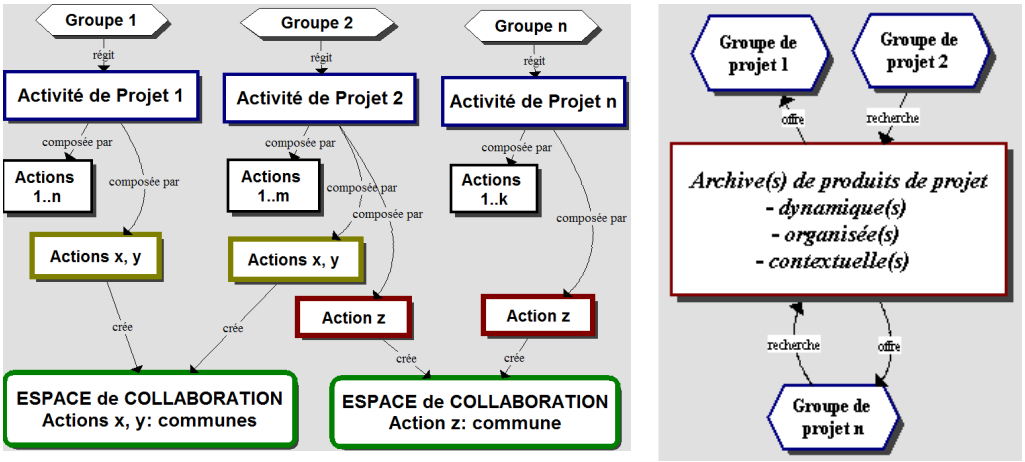


Figure I-1. (a) Espace de collaboration ; (b) Espace de documentation ;

Quant à l'espace dynamique de documentation – l'EDD – il est créé par la capitalisation de résultats de projet et leur utilisation appropriée (*cf.* Figure I-1, b). Les artefacts, comme les solutions documentées, les rapports ou les outils, construits ou transformés lors d'une activité de projet, cristallisent l'expérience de leurs utilisateurs et intègrent l'histoire de leur développement (Kuutti et Kaptelinin, 1997; Rabardel, 1995). Ils peuvent et ils doivent être mis à profit au sein de la communauté d'apprenants qui pourront ainsi avoir accès à des résultats de projet significatifs, puisqu'ils relèvent du même domaine d'apprentissage, sont issus des actions communes ayant des buts proches. Dans ce cas, l'EDD peut assurer la possibilité de confrontation des résultats des travaux entre apprenants ainsi que la prise en compte de l'histoire de projets afin de permettre aux apprenants de bénéficier de l'expérience vécue par d'autres.

1.1.2.2.2. *Construire l'EDC et l'EDD avec des ontologies*

Pour permettre aux apprenants de construire l'EDC ainsi que d'utiliser et d'enrichir l'EDD, le système ERPAD offre trois services de base (*cf.* Figure I-2). Le premier permet aux apprenants de décrire sémantiquement leurs actions et les produits qui en résultent, et cela, d'une manière standardisée, en utilisant une ontologie du domaine d'apprentissage et une ontologie de tâche. Ceci est un excellent moyen de normalisation de la sémantique utilisée pour décrire les buts et les actions ainsi que les ressources produites ou utilisées lors de la réalisation des projets, moyen qui constitue la pierre angulaire du processus d'identification des actions communes (2^{ième} service), point de départ de la construction émergente de l'EDC, et du processus d'enrichissement de l'EDD (3^{ième} service).

Le deuxième service met en contact les apprenants en identifiant leurs actions communes en fonction de la proximité sémantique existant entre les références des actions et des buts d'apprenants. La valeur de proximité est calculée par un agent de l'ERPAD en se basant sur des relations sémantiques existantes entre les concepts de l'ontologie ainsi que sur le nombre de concepts identiques (Rogozan, Hotte, et Abdulrab, 2002).

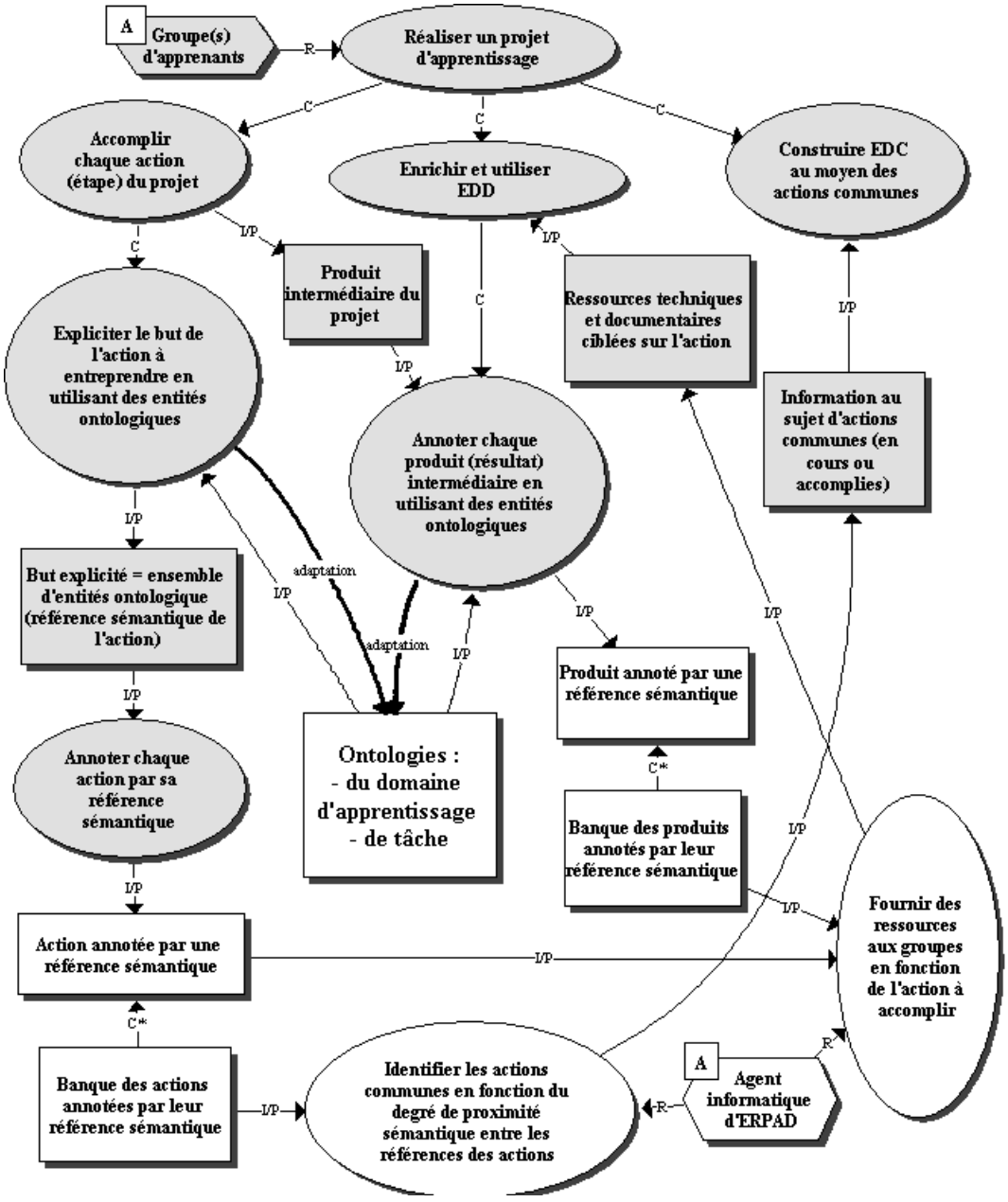


Figure I-2. Utilisation des ontologies pour la construction de l'EDC et de l'EDD

Finalement, le troisième service offre des fonctions d'archivage et d'annotation de ressources produites dans les projets ainsi que des fonctions de recherche « intelligente » mettant en relation les actions d'apprenants avec des ressources pertinentes, ciblées sur les buts d'apprenants.

1.1.2.3. **Évolution des ontologies pour un ERPAD co-évolutif**

Le fonctionnement du système d'apprentissage ERPAD s'articule principalement sur une boucle de co-évolution : (1) en utilisant des ontologies, ERPAD oriente l'attention des apprenants vers une définition des buts et des actions communes et vers la description sémantique des produits de projet, (2) pendant la réalisation des projets, les apprenants construisent leurs espaces de travail en utilisant les ressources fournies par l'ERPAD, et (3) ce processus de construction modifie les conditions environnementales de réalisation de projets, le comportement de système d'apprentissage, les ressources utilisées, les pratiques et les méthodes de travail, mais aussi, ou surtout, les ontologies du domaine et de la tâche.

Pourquoi les ontologies doivent-elles être modifiées lors des activités d'apprentissage supportées par l'ERPAD ? D'une part, parce que la construction émergente de l'EDC et de l'EDD nécessite une rétroaction évolutive : le référencement sémantique des nouvelles actions ou ressources pourrait demander la modification de leur référentiel sémantique afin de prendre en compte des nouvelles caractéristiques. D'autre part, les ontologies d'ERPAD ne peuvent pas être entièrement prédéfinies puisque l'apprentissage par projet est une démarche constructive et flexible, sans une structure de connaissances ou pédagogique complètement préétablie. À mesure que l'apprentissage se matérialise, qu'il passe de l'état de projet envisagé à l'état d'activité analysée, les connaissances des acteurs évoluent, ces derniers créant une compréhension du monde qui leur est propre, qui naît avec le projet et qui se construit pendant le projet. Cette compréhension pourrait demander la modification de l'ontologie du domaine si les acteurs considèrent qu'elle ne satisfait plus ses fonctions de référentiel sémantique. En outre, même si la question problème du projet peut être prédéfinie, la démarche pour y répondre et les actions pédagogiques mises en œuvre restent peu prévisibles.

Par conséquent, il ne suffit pas de rendre disponibles les ontologies dans l'ERPAD sans aucun moyen de les faire évoluer en fonction des besoins exprimés par les apprenants. Les ontologies, vues comme des structures immuables, ne peuvent tenir compte ni de l'évolution des connaissances d'apprenants, ni du développement de l'activité de projet. Il est indispensable alors d'offrir aux apprenants la possibilité d'adapter, pendant leur démarche de projet, l'ontologie du domaine d'apprentissage et celle de la tâche (Rogozan et Paquette,

2003). En conclusion, l'**évolution de l'ontologie est une condition *sine qua non* pour la conception appropriée de l'ERPAD**, celui-ci devant être capable de gérer cette évolution dans un environnement dynamique et distribué sur le web.

1.2 **Contexte de la thèse**

Dans la section précédente, nous avons mis en évidence la nécessité de donner aux apprenants des moyens méthodologiques et technologiques pour faire évoluer les ontologies du domaine et de la tâche dans le but de leur permettre de devenir les gestionnaires de leur propre ERPAD (Espace de Réalisation des Projets d'Apprentissage à Distance). ERPAD possède trois composants principaux : (1) les éléments pédagogiques (c.-à-d. acteurs, ressources et activités/actions de projets) ; (2) les ontologies utilisées comme des référentiels sémantiques ; (3) une infrastructure web pour la diffusion en ligne des événements d'apprentissage.

En prenant en compte ces trois composants de l'ERPAD, nous pouvons mettre en contexte nos travaux de recherche, tel qu'illustré dans la Figure I-3. L'association entre les ontologies, les ressources référencées sémantiquement et l'infrastructure web, connue actuellement sous le nom du « Web sémantique » (Berners-Lee, Hendler, et Lasilla, 2001), constitue notre contexte général de recherche. Les objectifs premiers du Web sémantique – (1) exprimer formellement la signification et le contenu des ressources ; (2) permettre une recherche « intelligente » des ressources en fonction des requêtes des agents humains ou informatiques ; (3) effectuer des raisonnements sur et avec ces ressources ; (4) faciliter l'interopérabilité sémantique entre systèmes et la communication entre humains – sont des éléments fondamentaux de support à la mise à distance des systèmes d'apprentissage distribués et multi-acteurs (Anderson et Whitelock, 2004). Notre contexte particulier est alors l'utilisation du Web sémantique comme infrastructure pour le déploiement des systèmes d'apprentissage en ligne, comme, entre autres, l'ERPAD, dont les composants nécessitent une description sémantique de leurs contenus et services, en vue d'obtenir un Web Sémantique Éducatif.

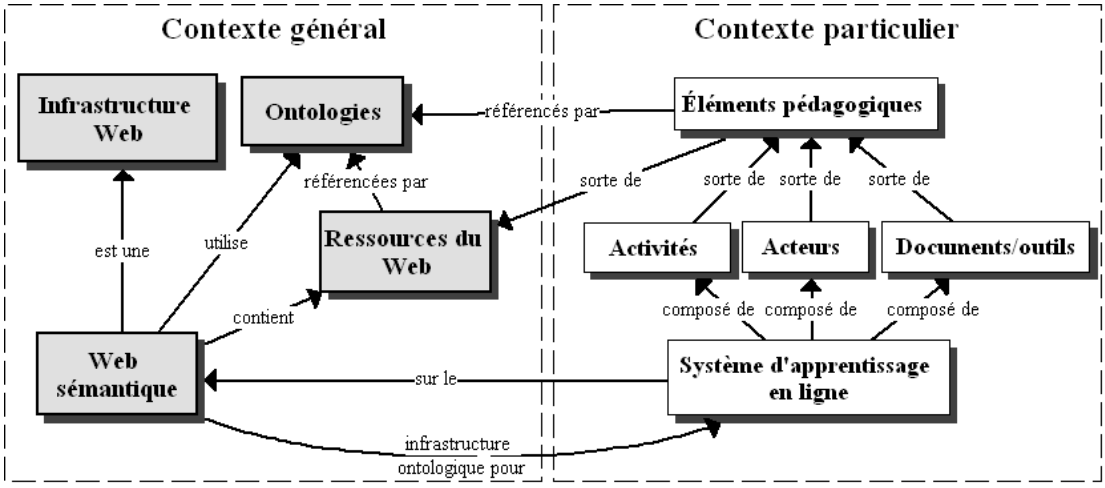


Figure I-3. Contexte de recherche : contexte général et contexte particulier

1.2.1. **Contexte général : le Web sémantique**

Une des visions à la base du Web sémantique est celle d'associer des connaissances formalisées, provenant d'ontologies, au contenu informel du web. Dans un premier temps, nous présentons donc ce qu'est une ontologie. Dans un deuxième temps, nous discutons la place des ontologies dans l'architecture du Web sémantique et les langages utilisés pour les formaliser.

1.2.1.1. **Ontologies en informatique**

En informatique, les ontologies sont apparues au début des années 90[9], à la suite des démarches d'acquisition des connaissances dans les systèmes à base des connaissances (SBC). Ces démarches proposaient de dégager les objets du domaine, leurs significations et leurs relations, des objets du raisonnement décrivant les règles heuristiques d'utilisation des objets du domaine. Le but était de faciliter la construction des SBC en permettant la réutilisation des bases génériques des connaissances (Neches et al., 1991; Schreiber, Wielinga, et Breuker, 1993). En ce sens, les chercheurs ont proposé de fonder ces

[9] Notons que l'origine des ontologies peut aller plus loin que les années 90. Au début des années 80, des travaux comme ceux de Brachman (1983) sur les réseaux sémantiques s'intéressaient déjà aux classifications des objets du monde.

connaissances sur la spécification d'une ontologie, définie comme l'ensemble, structuré par des relations principalement taxonomiques, des objets reconnus comme existant dans un domaine.

1.2.1.1.1. *Définition(s) du concept d'ontologie*

Pour avancer vers une définition complète du concept d'ontologie en informatique, revenons d'abord à un travail de Guarino (1997) et de Guarino et Giaretta, (1995). Les auteurs identifient une palette d'interprétations possibles, en allant de « l'ontologie comme système conceptuel informel », au niveau sémantique, jusqu'à « l'ontologie comme la représentation partielle d'un système conceptuel via une théorie logique et son vocabulaire », au niveau syntaxique. Parmi les autres définitions identifiées, on retrouve celle de Gruber (1995), pour qui « une ontologie est une spécification *explicite* d'une *conceptualisation* », ou encore celle de Borst (1997), qui affirme qu'une une ontologie est « la spécification *formelle* d'une conceptualisation *partagée* ».

Ces définitions peuvent être expliquées comme suit :

- **Conceptualisation**. Ce terme réfère à une vue du monde, la représentation d'un domaine de discours en termes d'**entités ontologiques**, dont l'organisation décrit la partie du monde du domaine conceptualisé (Gomez-Perez, 1999). Ces entités sont les concepts, les relations, les axiomes et les instances :
    - Les **concepts** (ou **classes**) référent à des entités, objets, phénomènes, processus, stratégies, raisonnements du domaine de discours.
    - Les **relations** (ou **propriétés**) représentent divers types d'interaction entre les concepts.
    - La **relation « is-a »** est une relation de subsomption spécifiant des liens de généralisation permettant l'héritage des propriétés entre les concepts.
    - Les **axiomes** explicitent des énoncés conceptuels toujours vrais dans le contexte de l'ontologie et sont utilisés pour contrôler la signification des concepts et des relations.

- Les **instances** sont utilisées pour représenter des objets concrets qui appartiennent à des concepts.

Plusieurs exemples de concepts/classes, relations/propriétés et axiomes sont présentés dans l'Appendice B où nous discutons de la représentation des ontologies à l'aide du langage OWL (*Web Ontology Language*).

- **Explicite/Formelle**. Ces termes réfèrent au fait que toutes ces entités ontologiques (concepts, relations, axiomes et instances) doivent être définies explicitement à l'aide d'un langage ayant une sémantique plus ou moins formelle. Ainsi, en fonction de son degré de formalisation (Uschold et Gruninger, 1996), une ontologie peut être informelle (en langage naturel), semi-informelle (en langage naturel structuré et limité), semi-formelle (en langage artificiel défini formellement) et formelle (exprimée par des termes précis, définis au moyen d'une syntaxe et d'une sémantique formelle comme la logique descriptive, par exemple).

- **Partielle**. Étant donné que la conceptualisation est généralement spécifiée d'une manière précise et formelle, l'ontologie ne peut pas assumer toute la richesse interprétative du domaine conceptualisé et ne le fait donc que partiellement.

- **Partagée**. Ce terme met en évidence le fait qu'une ontologie capture la connaissance consensuelle, non réservée à quelques individus, mais acceptée par un groupe.

En regroupant les définitions exposées ci-dessous, nous pouvons conclure que l'ontologie est la spécification, plus ou moins formelle, rendant compte partiellement, mais explicitement, d'une conceptualisation partagée au sein d'une communauté. Nous ne nous attardons plus sur ce qu'est une ontologie, ses caractéristiques étant largement discutées dans les articles précisés ci-dessus ainsi que dans (Chandrasekaran, Josephson, et Benjamins, 1999; Charlet, Bachimont, et Troncy, 2003).

1.2.1.2. **Web sémantique, langages et ontologies**

Le web actuel est essentiellement syntaxique. La structure des ressources est bien définie, mais leur contenu reste inaccessible aux traitements machines, seuls les humains

étant capables de l'interpréter. Le web sémantique a l'ambition de lever cette difficulté en associant aux ressources des concepts formels pour décrire leur contenu, le contexte de leur utilisation ainsi que les services fournis. Ceci permettra alors aux différents agents logiciels d'accéder et d'exploiter directement le contenu des ressources et de raisonner dessus. On verra alors les utilisateurs déchargés de leurs tâches de recherche, de combinaisons de ressources ou d'identification de services web pertinents (Hendler, 2001). On verra aussi résolus les problèmes d'interprétation des ressources provenant de sources hétérogènes et réparties. Ceci facilitera la communication entre les divers systèmes du Web et leur permettra d'offrir des services ciblés sur les besoins des utilisateurs (Wache et *al.*, 2001). On verra s'accomplir l'idée de l'espace virtuel de demain, telle que proposée initialement par Berners-Lee, Hendler et Lasilla (2001).

1.2.1.2.1. *Description formelle des ressources du Web Sémantique*

Afin de clarifier la terminologie utilisée, remarquons que deux termes sont principalement utilisés pour décrire la sémantique associée aux ressources du web – **métadonnée** et **annotation**. Les deux termes se réfèrent à une information structurée et descriptive, ajoutée à une ressource sous la forme des marqueurs exploitables informatiquement.

Dans le contexte du web sémantique, la différence entre métadonnées et annotations est presque insaisissable, la plupart des scientifiques les utilisant pour dénoter la description, explicite et formelle, du contenu d'une ressource à partir des éléments standards ou des connaissances disponibles dans une ou plusieurs ontologies. Certains auteurs soulignent cependant une différenciation, assez subtile, entre les métadonnées et les annotations. Pour Prie et Garlatti, (2004), une métadonnée est vue comme une « donnée sur une donnée », suivant un schéma standard comme le Dublin Core (DC)[10], par exemple ; elle sera plutôt l'information attachée à une ressource du web au moyen d'une représentation externe et aura ainsi une existence plus indépendante, voire même comme ressource en tant que telle. Quant à l'annotation, vue comme une remarque que l'on peut faire sur une ressource, elle sera plus

[10] Consultez le site http://dublincore.org/ pour des informations sur le schéma de métadonnées de Dublin Core.

située au sein de cette ressource et « rédigée » au cours d'un processus d'annotation. Remarquons que, pour le web sémantique, on parlera le plus souvent des « annotations sémantiques » au sens qu'elles apportent une information riche sur les ressources, mais en même temps, interprétable informatiquement (Bechhofer et Goble, 2001). Par exemple, les annotations sémantiques créées avec SHOE (Heflin, 2001), permettant d'insérer dans des pages web des marqueurs spécifiques, basés sur des ontologies.

Outre la distinction ci-dessus, on dégage également la notion de métadonnées objectives, provenant des schémas standards qui définissent *a priori* la sémantique des ressources utilisées dans un contexte précis, et celle d'annotations plus subjectives, représentant plutôt des points de vue sur les ressources, des points de vues qui dépendent subjectivement de l'auteur, selon Nilsson, Palmer, et Naeve, (2002). Dans le premier cas, l'utilisation d'un langage comme le XML (*eXtensible Markup Language*) est suffisante pour la formalisation. Le deuxième cas nécessite, par contre, une flexibilité du langage de description, par exemple comme celle offerte par le RDF (*Resource Description Framework*) dont le but est de définir un mécanisme pour décrire les ressources sans faire d'hypothèses sur un domaine particulier d'application.

Enfin, certains auteurs (Paquette et Rosca, 2004; Paquette, Rosca, Mihaila, et Masmoudi, 2007) tentent de concilier les termes métadonnées et annotations pour en arriver aux **références sémantiques** associées ou intégrées dans les ressources pour décrire leur contenu. Ces références sont principalement des descripteurs sémantiques choisis à partir des référentiels de formes diverses : métadonnées normalisées, taxonomies, ontologies formalisées. Étant donné la portée plus générique de cette dernière dénomination, nous allons l'utiliser par la suite au sens de description formelle du contenu de ressources.

1.2.1.2.2. *Architecture du Web Sémantique*

L'architecture du web sémantique (*cf.* Figure I-4) repose sur une hiérarchie des langages de définition d'ontologies et sur une hiérarchie des langages d'assertion, permettant d'associer des descripteurs sémantiques à des ressources. Elle repose également sur un ensemble de services pour accéder aux ressources en utilisant leurs références sémantiques,

pour gérer l'évolution et la gestion des versions d'ontologies et pour utiliser des moteurs d'inférences capables d'effectuer des raisonnements complexes.

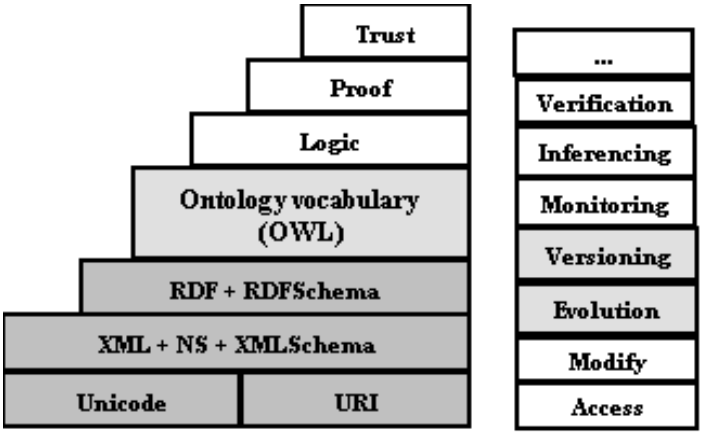


Figure I-4. Architecture du Web sémantique selon Berners-Lee, (2000), et Oberle, Volz, Motik et Staab, (2004)

L'architecture du web sémantique repose sur la technique de *Uniform Resource Identifier* (URI), qui permet l'attribution d'un identifiant unique à des ressources, en encodant l'information d'identification sous la forme d'une séquence des caractères séparés par des délimiteurs standardisés[11] (Berners-Lee, 2005). Le terme **ressource** est employé ici dans un sens général pour qualifier tout ce qui pourrait être identifié par un URI (p.ex. pages web, images, services, outils, personnes).

Le langage *eExtensible Markup Language* (XML) (W3C_Consortium, 2004) procure une syntaxe standard aux documents structurés. Il a l'avantage d'être conçu comme un métalangage qui permet à chaque langage du web de définir son propre format de document, d'écrire et d'utiliser des documents dans ce format. Il peut alors être vu comme la couche de transport syntaxique du web sémantique, tous les autres langages étant exprimables et échangeables dans la syntaxe XML. Mais le XML est limité puisqu'il ne dispose d'aucune sémantique permettant de décrire les ressources du Web de manière à les rendre intelligibles par des machines. C'est ainsi qu'est né le RDF.

[11] Par exemple, l'URI « `http://www.example.org/OntologieActeursTeleapprentissage` » identifie l'ontologie `OntologieActeursTeleapprentissage`.

Le langage *Resource Description Framework* (RDF) (W3C_Consortium, 2004) est un langage formel ayant une sémantique simplifiée permettant d'exprimer des relations entre les ressources du web en utilisant des triplets de la forme ressource–propriété–valeur, dont les éléments peuvent être des URI, des littéraux ou des variables. Ces triplets sont interprétables comme des graphes à arcs orientés et étiquetés, dont le nœud de départ est le sujet (ressource), l'étiquète de l'arc est le prédicat (propriété) et le nœud cible est l'objet (valeur). Au RDF s'ajoute *RDFschema* (RDFS) qui permet de déclarer des classes de ressources avec une sémantique pour définir des hiérarchies de généralisation (`rdfs:subClassOf`). RDFS permet aussi de déclarer des propriétés, définies comme des relations binaires entre les classes, en précisant leur domaine d'applicabilité (`rdfs:domain`) et leur domaine de valeurs (`rdfs:range`) ainsi que leur niveau de généralisation dans la hiérarchie de propriétés (`rdfs:subPropertyOf`).

Cependant, l'extension à RDFS ne fournit que des mécanismes primitifs pour spécifier les classes et les propriétés. Pour munir les langages du Web d'une sémantique formelle permettant la représentation des ontologies, les RDF et RDFS ont été enrichis par l'apport du langage ontologique OWL (*Web Ontology Language*) (Bechhofer et Goble, 2001) qui permet de définir des classes plus complexes en utilisant des constructeurs provenant de la logique de description (`intersectionOf`, `unionOf`, `complementOf`), des restrictions et des axiomes des classes (`equivalentClass`, `disjointWith`, `oneOf`) ainsi que des propriétés équivalentes, inverses, symétriques ou transitives. Pour permettre à tout lecteur de se familiariser avec ce langage, nous présentons dans l'Appendice B les trois sous-langages d'OWL (OWL-LITE, OWL-DL et OWL-Full), une synthèse des constructeurs OWL et des exemples[12] de leur utilisation.

Pour résumer, le XML-Schema fournit un cadre de structuration syntaxique qui peut être employé pour définir rapidement et sans ambiguïté un format de document (p.ex. document RDF ou OWL). Le RDF(S) permet de contrôler, le format des références sémantiques à l'aide des triplets RDF. Finalement, en utilisant un formalisme proposant des

[12] Tous les exemples s'inspirent de la conceptualisation du domaine du téléapprentissage, présentée dans (Paquette, 2002).

mécanismes de raisonnement et une possibilité d'exprimer des connaissances hiérarchisées, le langage OWL permet de décrire formellement le contenu de toute ressource, quel que soit son degré de complexité, à l'aide de références sémantiques qui lui sont associées. C'est ici la richesse du Web sémantique : le contenu de chaque ressource est décrit formellement pour que tout agent du web soit capable de l'interpréter et de raisonner dessus en utilisant des mécanismes fondés sur la logique descriptive.

1.2.1.2.3. *Ontologies du Web Sémantique*

Les ontologies du Web Sémantique sont formelles et sont décrites à l'aide d'un langage ayant une sémantique riche et consistante : le langage standard OWL.

> **Définition (Ontologie OWL)**. Une ontologie OWL est une collection d'informations incluant des descriptions de classes, des propriétés et des instances. Une classe décrit d'une façon formelle un concept abstrait du domaine de l'ontologie. Une instance est un membre de l'extension de la classe, cette extension étant définie comme l'ensemble des tous les individus concrets appartenant à cette classe. Une propriété est une relation binaire qui énonce des faits généraux au sujet des classes ou des faits spécifiques au sujet des instances. Une ontologie est complètement définie quand les concepts sont structurés par leurs propriétés et que ces propriétés sont combinées dans des axiomes permettant d'effectuer des inférences sur le contenu de l'ontologie.

1.2.2. **Contexte particulier : le Web Sémantique Éducatif**

Restreindre le Web sémantique à son architecture et ses langages serait limitatif : ce sont ses applications qui font vivre cette vision et qui y sont la preuve du concept. Les principales applications du Web sémantique visent, entre autres, le domaine du commerce électronique, les portails et mémoires d'entreprise, le traitement automatique des langues, la recherche d'informations, mais également le domaine de la formation en ligne. Ce dernier trouve ainsi un excellent moyen pour développer une approche plus globale de l'éducation fondée sur la réutilisation et le partage des ressources pédagogiques, mais aussi sur l'utilisation des agents logiciels capables d'interpréter le contenu de ressources et de l'adapter au profil des utilisateurs. Une discussion forte intéressante sur les impacts réels du Web

sémantique et des banques de ressources éducatives dans le contexte des communautés d'apprentissages et formation en ligne peut être consultée dans (Hotte et Contamines, 2007).

Dans la formation en ligne, le Web sémantique se présente comme une infrastructure prometteuse pour la mise en place de systèmes d'apprentissage distribués qui soient flexibles, adaptables aux besoins des utilisateurs et réutilisables dans plusieurs contextes d'apprentissage (Anderson et Whitelock, 2004; Koper, 2004; Stojanovic, Staab, et Studer, 2001; Stutt et Motta, 2004). En utilisant des ontologies pour décrire la sémantique des ressources interconnectées qui composent les cours en ligne, le Web sémantique facilite l'interopérabilité et la gestion des ressources provenant des sources hétérogènes et réparties ainsi que la réutilisation des modèles d'enseignement fondés sur des ontologies partagées par diverses communautés. Grâce au référencement sémantique des ressources pédagogiques ainsi qu'à la possibilité d'effectuer un appariement sémantique entre ces références et diverses requêtes, le Web sémantique accroît la capacité des moteurs de recherche à offrir des réponses adaptées aux besoins des utilisateurs. Enfin, il peut supporter la création dynamique d'espaces de télétravail personnalisés, étant donné qu'il permet d'apparier les préférences, les objectifs ou les buts des acteurs avec des éléments pédagogiques distribués (p.ex. des objets pédagogiques, des activités de télétravail ou des facilitateurs de l'apprentissage). Supportée par les nouvelles technologies sémantiques du Web, la formation en ligne s'aligne alors vers le paradigme émergeant du Web sémantique éducatif (Anderson et Whitelock, 2004; Aroyo et Dicheva, 2004).

TELOS (TELelearning Operating System) est un excellent exemple d'application du Web sémantique éducatif (Paquette, Rosca, Mihaila, et Masmoudi, 2007). Développé dans le cadre du projet LORNET, ce système vise à favoriser la réutilisation des ressources pédagogiques et des modèles de *workflow* pédagogique qui sont référencés sémantiquement, en utilisant diverses ontologies. Par cette description sémantique, le but de TELOS est d'offrir, en plus des mécanismes de « recherche intelligente », des mécanismes d'assemblage et de synchronisation de ressources et de *workflow*s, ou encore des fonctions d'assistance, qui permettront aux utilisateurs d'opérer efficacement aussi bien dans un environnement prédéfini que dans un environnement émergeant contrôlé et transformé par les utilisateurs (Paquette et Rosca, 2002). Nous trouverons ainsi en TELOS une plate-forme ouverte dont les

capacités d'organisation et d'agencement flexible des éléments référencés sémantiquement nous offrent le contexte approprié d'implémentation de l'espace de réalisation des projets d'apprentissage à distance (ERPAD).

ERPAD est un système d'apprentissage du Web sémantique, composé d'un ensemble d'éléments pédagogiques (activités et ressources de projet, acteurs de l'apprentissage) combinés pour former un tout cohérent. Le référencement sémantique des éléments pédagogiques à un même espace de connaissances crée un environnement où la combinaison physique des éléments est accompagnée d'une agrégation de leur sens. De cette manière, tout agent peut avoir une compréhension intégrale des processus éducationnels qui se développent dans l'ERPAD et peut donc l'observer et l'influencer. Ce moyen d'agrégation sémantique est d'autant plus important pour un participant qui aborde ce processus de l'intérieur (p.ex. les apprenants qui construisent, d'une manière participative, un espace de collaboration et de documentation) puisque toute intervention pertinente suppose une synchronisation sémantique, c'est-à-dire le partage d'un même sens avec les autres participants.

1.3 **Problématique et objectifs de la thèse**[13]

1.3.1. **Présentation de la problématique de recherche**

Un des aspects les plus importants de l'utilisation des ontologies renvoie à leur caractère évolutif, et cela, tant pour les ontologies du Web sémantique que pour celles utilisées dans un système d'apprentissage du Web sémantique.

Dans les systèmes d'apprentissage du Web sémantique, les ontologies sont principalement utilisées comme sources de référencement sémantique pour décrire le contenu des ressources, les objectifs et l'enchaînement d'activités pédagogiques et de compétences des acteurs. Comme ces systèmes ne sont pas des entités fixes – ils sont réorganisés, modifiés, fusionnés ou fractionnés pour répondre aux besoins changeants de diverses

[13] D'une manière plus brève, nous avons déjà présenté le questionnement et les objectifs de recherche dans le chapitre d'introduction. Nous considérons cependant qu'un rappel des éléments de problématique, après avoir illustré le contexte de recherche, permettra au lecteur une meilleure compréhension des enjeux de la thèse.

démarches d'apprentissage – les ontologies seront aussi soumises à la modification. Plus particulièrement, dans un environnement d'apprentissage de type ERPAD, la modification du référentiel sémantique composé d'ontologies du domaine et de la tâche est indispensable en raison du caractère dynamique et non prédéfini de l'apprentissage par projet.

D'un point de vue plus général, les ontologies du Web sémantique doivent évoluer (Charlet, Bachimont, et Troncy, 2003; Ding, Fensel, Klein, Omelayenko, et Schulten, 2004; Euzenat et all., 2001; Heflin et Hendler, 2000; Klein et Fensel, 2001; Maedche, Motik, et Stojanovic, 2003; McGuinness, 2000; Noy et Klein, 2003; Stojanovic et Motik, 2002; Studer, 2003). Étant donné que le Web croît et change constamment, nous devons nous attendre à ce que les ontologies changent aussi. Les ontologies doivent changer afin de répondre aux modifications apparues dans le domaine conceptualisé ou parce qu'une nouvelle façon de modéliser le domaine apparaît plus indiquée. Différents agents du web peuvent avoir besoin de différentes vues d'une même ontologie, ce qui exige l'adaptation de l'ontologie pour refléter ces vues. Ou encore, en raison du caractère multi-acteurs du web, on peut difficilement supposer que la conceptualisation explicitée par les ontologies ne sera aucunement soumise aux modifications lors du processus d'échange d'informations et de significations entre les acteurs.

Bien qu'assez récents dans la communauté scientifique (apparus au début 2000), les travaux sur l'évolution de l'ontologie sont essentiels pour le Web sémantique (W3C_WebOnt, 2004a; Web_sémantique, 2001). La conclusion est la suivante :

> **Identification de la problématique de recherche.** Les ontologies du Web sémantique doivent évoluer, les ontologies utilisées dans un système d'apprentissage du Web sémantique (et particulièrement dans un système de type ERPAD) doivent aussi évoluer. Des méthodes et des outils doivent être conçus pour assurer l'évolution des ontologies. Cela nous amène à identifier notre problématique de recherche, à savoir : **comment supporter l'évolution d'une ontologie du Web sémantique tout en préservant l'intégrité du référencement sémantique des ressources** ?

1.3.2. **Identification d'objectifs et contributions de recherche**

1.3.2.1. **Objectifs de recherche**

Un certain nombre de questions fondamentales, liées à l'évolution des ontologies, doivent être posées. Par exemple, doit-on s'intéresser aux changements simples, comme l'ajout ou l'effacement des composants d'une ontologie, ou également aux changements plus complexes, comme la fusion ou la division, qui possèdent une sémantique plus riche pour exprimer l'évolution ? Les utilisateurs d'une ontologie doivent-ils ou non être informés de changements apportés lors de l'évolution ? Si oui, par quel moyen les identifier et comment les présenter pour en faciliter la compréhension des utilisateurs ? Ou encore, comment s'assurer qu'après l'évolution des ontologies, les ressources référencées restent accessibles et que les traitements fondés sur les ontologies continuent à bien fonctionner ?

Comme tout processus complexe, l'évolution nécessite un cadre méthodologique qui la structure. Les propositions méthodologiques actuelles (OntoWeb, 2002a) traitent de la construction des ontologies d'une manière assez poussée. Elles sont pourtant loin d'aborder l'évolution des ontologies dans son ensemble et ne fournissent non plus des recommandations ou des méthodes pour l'orienter. Par ailleurs, du peu d'approches qui parlent de l'évolution (Ehrig et al., 2003), aucun consensus ne se dégage. Ceci est dû particulièrement à une perspective différente sur ce qu'est l'évolution. Dans un contexte centralisé, elle signifie un processus d'application des changements et l'utilisation de cette seule ontologie évoluée (Stojanovic, 2004). Dans un contexte décentralisé, elle est davantage considérée comme un processus permettant l'identification et la gestion des versions multiples d'une même ontologie (Klein, 2004). Dans ces circonstances, nous énonçons le premier objectif de cette thèse, un objectif qui vise l'aspect conceptuel de la thèse :

> **1. Méthodologie d'évolution d'ontologies (objectif sur le plan conceptuel)**. Notre premier objectif de recherche est de concevoir une méthodologie présentant une vue unifiée sur l'évolution, sur son déroulement, sur ses exigences et sur ses besoins, sur les principes mis en jeu et sur les règles à respecter lors de l'évolution des ontologies dans un environnement dynamique et distribué sur le Web sémantique (*cf.* Chapitre II).

Nous devançons la présentation des résultats de cet objectif, en soulignant que nous avons organisé la méthodologie d'évolution selon deux axes. Le premier est la **phase d'évolution** consacrée aux changements et à la manière de les appliquer tout en préservant la consistance de l'ontologie. Le deuxième axe est la **phase de mise en opération de l'évolution**, consacrée au maintien de l'intégrité de ce qui repose sur l'ontologie, comme le référencement sémantique de ressources, par exemple. Pour qu'une méthodologie soit fonctionnelle, elle doit être supportée technologiquement. Cependant, comme nous allons le montrer dans le paragraphe 2.3.1.2. bien que la phase d'évolution soit plus ou moins desservie par des outils actuels, celle de la mise en opération de l'évolution, et particulièrement pour les ontologies utilisées comme référentiels sémantiques de ressources, est loin d'être supportée par ces outils. Nous arrivons ainsi à identifier les deux autres objectifs qui visent, tous les deux, l'aspect plus technologique de cette thèse.

2. Support à la gestion de l'historique de changements (objectif sur le plan conceptuel et technologique). Notre deuxième objectif est celui de proposer une méthode et un outil (*ChangeHistoryBuilder*) pour tracer l'historique des changements élémentaires et complexes, apportés à une ontologie, afin de permettre leur identification à tout moment après l'évolution de l'ontologie (*cf.* Chapitres III et IV).

3. Support à la gestion du référencement sémantique (objectif sur le plan conceptuel et technologique). Notre troisième objectif est celui de proposer une méthode et un outil (*SemanticAnnotationModifier*) pour analyser les effets des changements sur le référencement sémantique de ressources (en termes d'accès et d'interprétation de ressources) et pour en tenir compte afin de permettre une modification du référencement en fonction des changements apportés aux ontologies (*cf.* Chapitre V).

1.3.2.2. **Contributions apportées**

Si les travaux sur les ontologies, leur construction et leur utilisation, ont déjà une dizaine d'années, ceux qui concernent l'évolution de l'ontologie sont bien plus récents et leur poursuite est indispensable pour que les ontologies servent le Web sémantique et tout

système d'apprentissage du Web sémantique. Dans ce contexte, nous envisageons notre contribution sur deux plans :

1. **Sur un plan conceptuel**. Nous proposons une méthodologie qui (1) unifie les approches méthodologiques de l'évolution de l'ontologie, actuellement presque inexistantes et (2) intègre des éléments méthodologiques novateurs, principalement en ce qui concerne la gestion des changements apportés à l'ontologie ainsi que la mise en opération de l'ontologie, une fois évoluée, dans le contexte du Web sémantique. Pour ce deuxième aspect, nous proposons alors de nouvelles méthodes bien précises afin de soutenir : la journalisation des changements d'une manière normalisée et riche sémantiquement, l'analyse de l'effet de changements sur le référencement sémantique des ressources distribuées et finalement, une méthode pour maintenir la consistance du référencement après l'évolution d'une ontologie.

2. **Sur un plan technologique**. Nous proposons deux prototypes informatiques pour supporter la phase de mise en opération de l'ontologie – le *ChangeHistoryBuilder* et le *SemanticAnnotationModifier*. Ces prototypes possèdent des fonctionnalités permettant une analyse sémantique du processus d'évolution ainsi qu'un référencement sémantique évolutif.

1.3.3. **Organisation de la recherche**

Dans cette thèse, nous avons utilisé une démarche de recherche exploratoire, axée sur l'élaboration de solutions conceptuelles novatrices, implémentées dans des réalisations informatiques de petite taille. Le but était de permettre l'enrichissement des solutions conceptuelles ou l'affinement de celles déjà implémentées ainsi que d'apporter la preuve que la conception et l'implémentation partielles sont sur la bonne voie. Dans la Figure I-5, nous illustrons l'organisation de la recherche, en lui faisant correspondre les chapitres qui présentent nos solutions conceptuelles ainsi que nos implémentations partielles. Pour valider les propositions avancées, nous avons mis en place une expérimentation axée sur l'utilité des solutions et des outils développés.

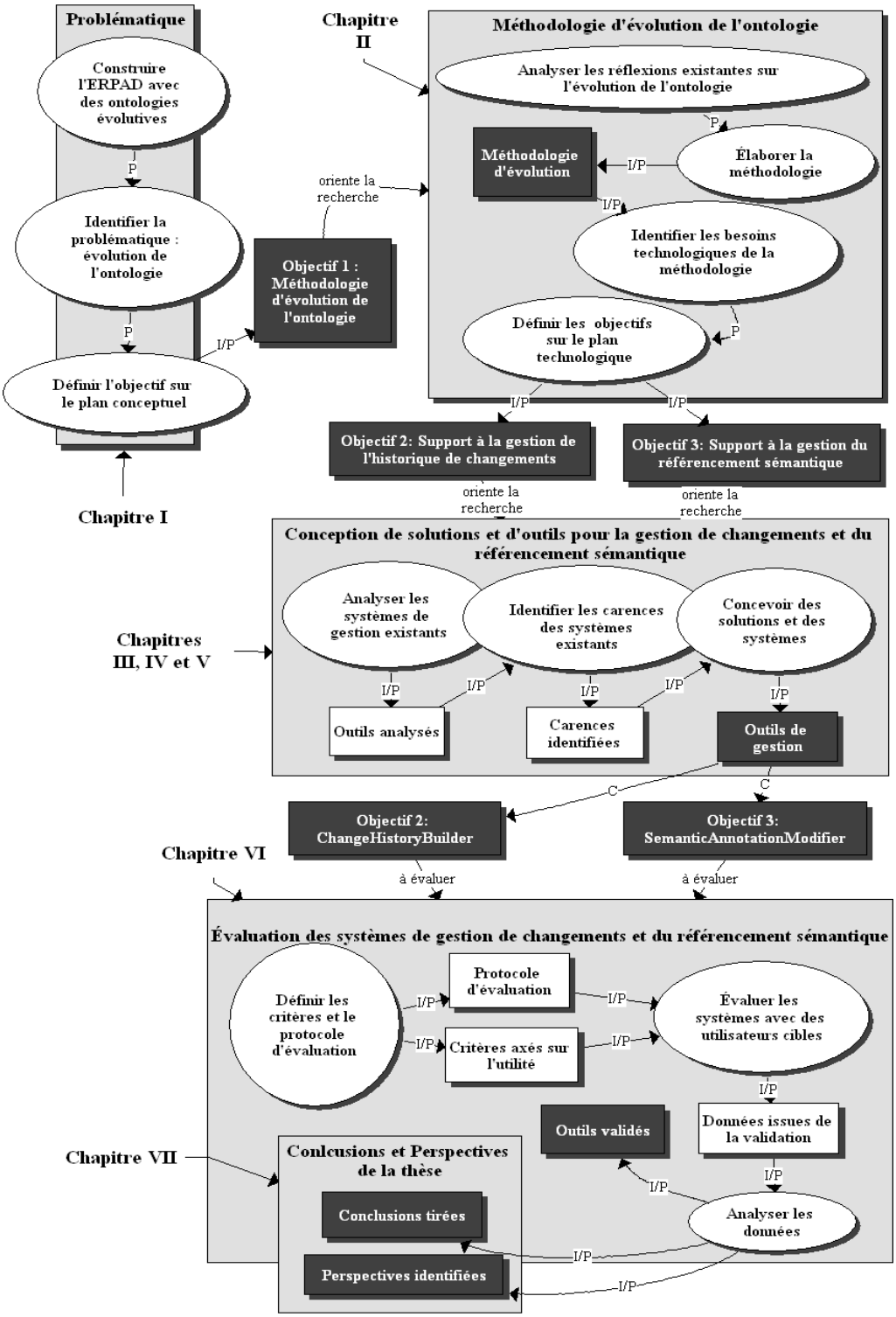


Figure I-5. Organisation de la recherche exploratoire
(La légende du formalisme graphique MOT est consultable dans l'Appendice A)

1.4 **Conclusion**

Dans un premier temps, nous avons fait le point sur nos travaux antérieurs ayant largement contribué à la genèse de la thèse. Ces travaux ont principalement mis en évidence la notion d'un espace de réalisation de projets d'apprentissage à distance (ERPAD), dont l'architecture est fondée sur des ontologies évolutives et sur le référencement sémantique des ressources d'apprentissage. Au cœur de l'ERPAD se trouve la gestion de l'évolution des ontologies, mais aussi la gestion du référencement sémantique qui repose sur ces ontologies évolutives. Ce qui nous a amenée à identifier notre problématique de recherche, à savoir : quelles pratiques et quels outils sont nécessaires pour supporter l'évolution des ontologies et pour maintenir l'intégrité du référencement sémantique des ressources après l'évolution de l'ontologie de référence.

Dans un deuxième temps, nous avons décrit le contexte général de la thèse qui est celui des ontologies et du Web sémantique, ainsi que le contexte, plus particulier, des systèmes d'apprentissage du Web sémantique. Ensuite, nous avons abordé plus en détail les questions soulevées par notre problématique de recherche. Nous avons présenté les trois objectifs de recherche que nous allons poursuivre dans le chapitre II, où nous proposons une méthodologie d'évolution des ontologies, dans le chapitre III, où nous présentons une ontologie des changements, dans le chapitre IV où nous illustrons une technique pour journaliser les changements lors de l'évolution, et dans le chapitre V, où nous développons une technique pour l'analyse et la résolution des effets de changements sur le référencement sémantique des ressources. Nous avons conclu ce chapitre en illustrant la démarche de recherche exploratoire utilisée dans la thèse.

Chapitre II

ÉVOLUTION DE L'ONTOLOGIE : MÉTHODOLOGIES ET OUTILS

Dans ce chapitre, nous présentons d'abord un état de l'art sur les approches méthodologiques d'évolution des ontologies et nous démontrons qu'elles n'arrivent pas à constituer une méthodologie complète (*cf.* Section 2.1). Par conséquent, nous énonçons notre objectif sur le plan conceptuel, soit l'intégration de ces approches dans une méthodologie unifiée, décrivant un processus complet d'évolution des ontologies.

Nous discutons ensuite cette méthodologie (*cf.* Section 2.2) et concluons par une analyse des outils ontologiques (*cf.* Section 2.3). La conclusion qui se dégage de cette analyse est, qu'actuellement, peu d'outils offrent des fonctionnalités pour supporter l'évolution, notamment pour supporter la journalisation de changements apportés aux ontologies ou pour supporter la maintenance de l'intégrité du référencement sémantique qui repose sur des ontologies évolutives. Pour combler ces lacunes, nous proposons un cadre de référence pour la gestion de changements apportés aux ontologies, cadre composé de deux modules *ChangeHistoryBuilder* et *SemanticAnnotationModifier*.

Les travaux de ce chapitre ont fait le sujet de deux publications : (Rogozan, 2004a; Rogozan, Paquette, et Rosca, 2004).

2.1 **Méthodologie pour l'évolution de l'ontologie : état de l'art**

Il existe actuellement une vaste palette d'approches[14] pour la construction des ontologies (OntoWeb, 2002a). Quelques approches décrivent la construction d'une ontologie par la fusion des ontologies différentes, par exemple les méthodes FCA-Merge (Stumme, 2001)ou PROMPT (Noy et Musen, 2000). Mais la plupart d'entre elles décrivent la construction d'une ontologie à partir de zéro, par exemple les méthodes SENSUS (Swartout, Ramesh, Knight, et Russ, 1997) ou KACTUS (Bernaras, Laresgoiti, et Corera, 1996), la méthode de (Uschold et King, 1995) ou celle de (Gruninger et Fox, 1995), ou encore les méthodologies METHONTOLOGIE (Fernandez, Gomez-Perez, et Juristo, 1997) et On-To-Knoweldge (Staab, Studer, Schnurr, et Sure, 2001).

Parmi toutes ces approches, seulement METHONTOLOGY et On-To-Knowledge considèrent l'évolution dans le cycle de vie d'une ontologie, mais elles ne fournissent ni méthodes, ni recommandations pour la supporter. Pourtant, plusieurs recherches mettent en évidence l'importance majeure de l'évolution des ontologies, surtout dans le contexte du Web sémantique (Charlet, Bachimont, et Troncy, 2003; OntoWeb, 2002a; WebOnt, 2004). Deux sont les approches le plus connues à l'heure actuelle qui traitent de l'évolution des ontologies sur le plan méthodologique : (1) l'approche développée par l'équipe de l'AIFB (Institute of Applied Informatics and Formal Description Methods) de l'université de Karlsruhe, et (2) l'approche développée par l'équipe du département IMSE (Information Management and Software Engineering) de l'université d'Amsterdam.

2.1.1.1. **Méthodologie de l'AIFB pour supporter l'évolution des ontologies**

L'évolution des ontologies est définie par Maedche, Motik et Stojanovic, (2003), et par Stojanovic, (2004), comme étant :

[14] Une méthodologie est une succession, compréhensible et intégrée, des méthodes et des techniques qui créent ensemble un système théorique explicatif pour l'accomplissement d'une tâche cognitivement complexe (IEEE, 1995). Une méthode est un processus ordonné utilisé pour l'ingénierie d'un produit ou pour exécuter un service. Une technique est une procédure utilisée pour atteindre un objectif donné.

Définition 1. La modification appropriée d'une ontologie et la propagation des changements dans les autres ontologies qui en dépendent.

Pour supporter l'évolution des ontologies, les auteurs proposent une méthodologie en trois étapes principales :

Représentation des changements. Cette étape vise l'édition des changements élémentaires, composites ou complexes. Dans le chapitre III nous présentons ce que les auteurs comprennent par un changement élémentaire, composite ou complexe et nous nous positionnons par rapport à leurs définitions.

Sémantique des changements. L'ontologie doit évoluer d'un état consistant vers un autre état consistant, c'est-à-dire un état où les contraintes du modèle ontologique sont respectées. Afin de résoudre les inconsistances introduites par certains changements, d'autres changements peuvent être nécessaires,. Cette étape vise alors de permettre la résolution de tous les changements additionnels[15] d'une manière systématique.

Propagation des changements. Le but de cette étape est de modifier automatiquement les instances et les ontologies dépendantes afin de préserver leur consistance avec l'ontologie évoluée. Pour cela, les auteurs proposent la modification des ontologies dépendantes par l'application récursive du processus d'évolution en fonction des changements apportés à l'ontologie évoluée.

2.1.1.2. **Limites de la méthodologie de l'AIFB**

Premièrement, les auteurs ne proposent aucune étape d'analyse des effets des changements sur la relation entre l'ontologie évoluée et les artefacts dépendants. Ceci est une limite importante étant donné que l'évolution de l'ontologie peut provoquer la détérioration du référencement sémantique des ressources, de l'interopérabilité avec d'autres ontologies ou encore la détérioration des fonctionnalités de systèmes qui utilisent une ontologie évolutive (Heflin, Hendler, et Luke, 1999; Stuckenschmidt et Klein, 2003a). En ce sens, l'étape de propagation des changements, proposée par les auteurs, est assez unidirectionnelle vu qu'elle

[15] Des exemples des changements additionnels sont présentés dans le paragraphe 2.2.1.

vise seulement la modification des ontologies dépendantes afin de préserver leur consistance avec l'ontologie de base.

Deuxièmement, les auteurs développent un processus d'évolution qui ne prend pas en compte la gestion des versions multiples. Ceci peut s'avérer convenable dans un environnement contrôlable où l'utilisation d'une ontologie peut être totalement prédéfinie. Toutefois, prédéfinir complètement l'utilisation d'une ontologie n'est pas une tâche possible sur le Web sémantique et, sans savoir comment une ontologie est utilisée, on peut difficilement propager systématiquement les changements dans les artefacts dépendants.

2.1.2. **Méthodologie de l'IMSE pour supporter la gestion des versions d'ontologie**

L'évolution des ontologies est définie par Klein, (2004), Noy, (2004), et par Noy, Kunnatur, Klein et Musen, (2003), comme étant :

> **Définition 2**. La capacité de gérer les changements apportés lors de l'évolution en créant et en maintenant différentes versions d'une ontologie. Cette capacité consiste à identifier et à différencier les versions, à modifier les versions, à spécifier des relations qui rendent explicites les changements effectués entre les versions.

Contrairement à la première définition qui considère l'évolution comme le passage d'un état de l'ontologie à un autre, en ne préservant que ce dernier, la deuxième définition considère plutôt la présence des plusieurs états d'une même ontologie (les versions) et la possibilité de les différencier. Les auteurs utilisent le terme versionnage pour décrire leur approche et proposent deux éléments fondamentaux pour une méthodologie de versionnage des ontologies :

- un modèle d'analyse de la relation entre deux versions d'une ontologie. Ce modèle permet (1) d'expliciter ce qui a été changé dans la définition des entités ontologiques, (2) de spécifier la relation sémantique entre les entités ontologiques dont la définition a été changée d'une version à une autre (p.ex. entités équivalentes ou conceptuellement différentes), (3) de décrire les changements par un ensemble de

métadonnées spécifiant la date, l'auteur et le but de chaque changement, (4) de décrire le contexte où les changements sont valides.

- une technique d'identification des versions d'une ontologie reposant sur deux principes de base : (1) les changements dans la définition d'une classe ou propriété produisent une nouvelle version, ayant un nouvel URI et (2) la forme d'URI indique si la version est compatible avec la version antérieure.

2.1.2.1. **Limites de l'approche de l'IMSE**

Les auteurs ne proposent pas une approche pour supporter l'évolution des ontologies, mais plutôt pour supporter la gestion des versions d'une ontologie après son évolution. Ils fournissent un modèle d'analyse de la relation entre les versions de l'ontologie, mais ne se préoccupent pas de la gestion de l'accès aux artefacts dépendants (les ressources référencées, les autres ontologies ou applications). De plus, les auteurs ne développent aucun cadre fonctionnel pour intégrer la totalité des éléments méthodologiques qu'ils proposent.

2.1.3. **Autres éléments méthodologiques pour l'évolution des ontologies**

La recherche actuelle sur les méthodologies d'évolution ne se résume pas toutefois uniquement à l'approche de l'AIFB et celle de l'IMSE. D'autres auteurs développent des éléments méthodologiques à cet effet.

Heflin et Hendler, (2000), développent SHOE, un langage fondé sur HTML, qui offre des primitives pour la gestion des versions multiples, en permettant d'associer à chaque version d'une ontologie un identifiant unique ainsi qu'une balise `Backward-Compatible_With` spécifiant les versions compatibles ou rétrocompatibles. Oliver, Shahar, Musen, et Shortliffe, (1999), définissent un modèle conceptuel, CONCORDIA, pour la gestion des changements d'une terminologie médicale. Ce modèle associe à chaque classe un identifiant unique, les classes pouvant ensuite être seulement retirées et non pas physiquement effacées. Ainsi, pour chaque classe, le modèle CONCORDIA est capable de garder la trace de toutes ses classes-parent ou enfants retirées en utilisant leur identifiant.

Enfin, McGuinness, (2000) fournit des recommandations théoriques pour garantir un processus d'évolution avec un minimum d'erreurs.

Du point de vue de l'évolution des ontologies dans un environnement multi-acteurs, Pinto, Staab et Tempich, (2004), proposent un modèle pour gérer la négociation des changements apportés par des acteurs distants qui tentent de modifier une ontologie partagée. Dans ce modèle, chaque utilisateur modifie l'ontologie à partir de son poste individuel, en construisant ainsi sa propre ontologie locale. Un comité scientifique analyse ensuite les ontologies locales pour identifier les changements à introduire dans l'ontologie partagée. Une approche similaire est développée par Sunagawa, Mizoguchi, et *al*., (2003), la différence étant que les acteurs distants modifient l'ontologie partagée sans aucun intermédiaire. Pour assurer la gestion de la dépendance entre les changements effectués, les différents acteurs peuvent choisir d'accepter les changements des autres, de réduire la dépendance par rapport aux changements des autres ou de rompre la dépendance.

Plus récemment, Xuan, Bellatreche et Pierra, (2006), proposent une approche formelle et un modèle de gestion des évolutions asynchrones des ontologies locales, qui peuvent évoluer indépendamment les unes des autres, mais qui référencent une seule et unique ontologie partagée. L'hypothèse à la base du travail de ces chercheurs est qu'une évolution d'une ontologie ne peut infirmier un axiome antérieurement vrai. Flouris, Plexousakis et Antoniou, (2006), développent, eux aussi, une théorie formelle et logique, fondée sur la notion de changement de croyance (*belief change*), pour résoudre le problème de conformité de représentation des connaissances dans le cas de l'évolution des ontologies OWL.

2.2 **Ébauche d'une méthodologie unifiée**

Bien que les approches présentées auparavant présentent des éléments méthodologiques pour supporter l'évolution des ontologies, elles ne fournissent ni une méthodologie complète ni un cadre intégrateur pour les éléments qu'elles proposent. Par conséquent, nous énonçons notre objectif conceptuel de recherche qui consiste : (a) à intégrer ces approches dans une méthodologie unifiée, décrivant un processus complet d'évolution des ontologies incluant la gestion des versions distribuées sur le Web sémantique ; (b) à

analyser des outils existants en fonction des étapes de la méthodologie afin d'identifier les besoins technologiques qui se font ressentir.

2.2.1. **Évolution des ontologies – définition**

Notre objectif d'unifier les deux approches méthodologiques dans un cadre cohérent requiert une définition qui englobe, en partie, les deux autres précédentes. En nous basant sur les travaux mentionnés ci-dessus[16], nous énonçons notre compréhension de l'évolution des ontologies :

> **Définition (Rôle de l'ontologie).** Le rôle de l'ontologie signifie le service assuré par l'ontologie, le but de son utilisation. Par exemple, l'ontologie peut servir comme source pour le référencement sémantique de diverses ressources d'apprentissage en ligne, ou encore elle peut servir comme armature pour la construction des systèmes 'intelligents' dont le comportement est guidé par l'ontologie.

> **Définition (Consistance de l'ontologie).** Une ontologie est consistante si les contraintes concernant le modèle structural et axiomatique de l'ontologie sont respectées. Les contraintes du modèle sont les caractéristiques structurales prédéfinies que l'ontologie doit respecter (p.ex. la hiérarchie des classes est un graphe taxonomique et acyclique), tout changement qui les invalide étant inconsistant (p.ex. déclarer la classe A comme sous-classe de B.1, tout en sachant que B.1 fait déjà partie des sous-classes de A). De même pour les axiomes de l'ontologie, un changement inconsistant est celui qui déclare une classe C comme sous-classe de A et B, tout en sachant que la classe A est disjointe de la classe B (*cf.* Figure II-1).

[16] Particulièrement, les travaux mentionnés dans les sections 2.1.1.1. et 2.1.2.

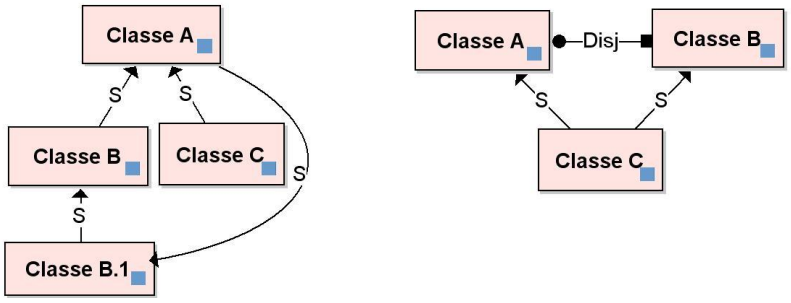


Figure II-1. Changement inconsistant : (a) par rapport au modèle ontologique ; (b) par rapport aux axiomes de l'ontologie.

(La légende du formalisme graphique MOT+Onto est consultable dans l'Appendice C)

Définition (Évolution de l'ontologie). Le processus de changement de l'ontologie en fonction des modifications dans son domaine, dans sa conceptualisation ou dans son usage. Ce processus se traduit en réalité par l'exécution d'un ou plusieurs changements pour passer d'une version de l'ontologie (V_N) à une version nouvelle (V_{N+1}), tout en préservant la consistance et les rôles de l'ontologie, mais aussi tout en assurant une gestion de l'histoire de changements apportés.

Définition (Changement élémentaire/complexe) Un changement se réfère à une modification de l'ontologie effectuée lors du processus d'évolution de V_N à V_{N+1}. Ces changements peuvent être élémentaires ou complexes. Un **changement élémentaire** est un changement simple et indécomposable, à la base de tout processus d'évolution, c'est-à-dire l'ajout ou l'effacement de composantes d'une ontologie. Un **changement complexe** est une collection ordonnée de changements élémentaires qui forment ensemble une seule et unique entité logique, dont la signification est bien définie et identifiée. À titre d'exemple nous indiquons le changement de fusion de classes, illustrée dans la Figure II-2 (nous donnons plus de détails sur les changements élémentaires et complexes dans le chapitre III).

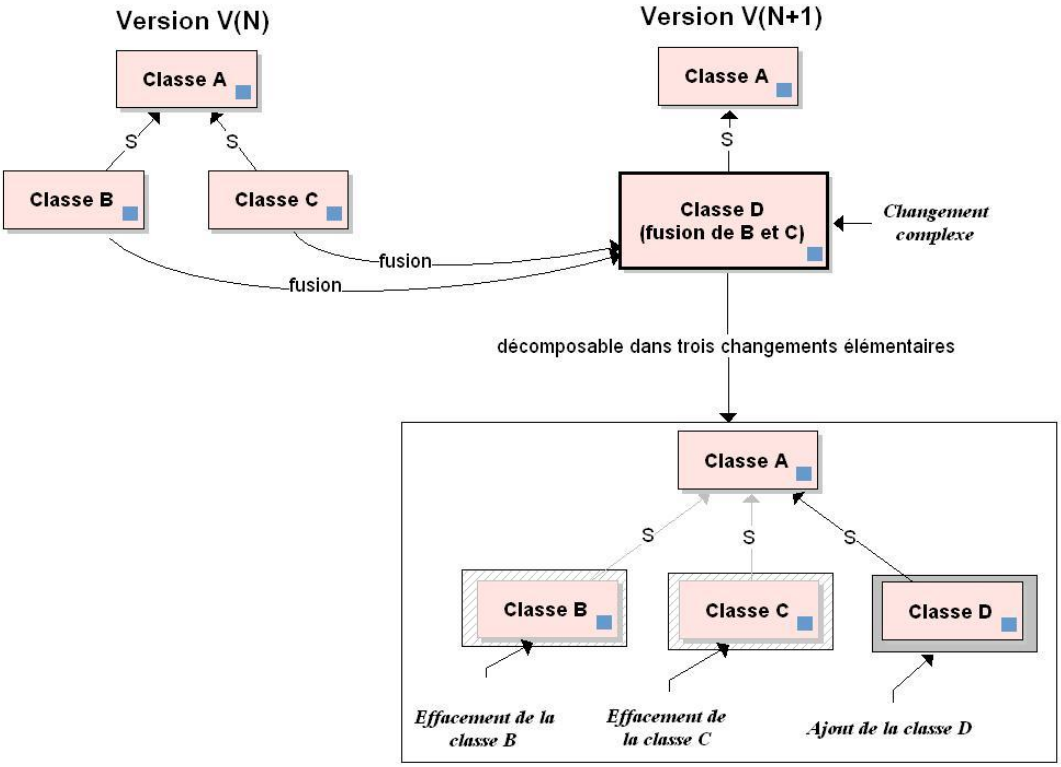


Figure II-2. Changement complexe de fusion (décomposable en trois changements élémentaires)

(La légende du formalisme graphique MOT+Onto est consultable dans l'Appendice C)

Définition (Changement primaire/additionnel). Pour préserver la consistance de l'ontologie, un changement peut en demander d'autres. Ainsi, un **changement primaire** est celui qui n'est la conséquence d'aucun autre changement. Un **changement additionnel** est celui qui est causé par un autre changement (dit changement-parent). Dans la Figure II-3, nous illustrons un exemple de changement primaire (la fusion des classes) et ses changements additionnels (le transfert de deux sous-classes et d'une propriété), d'autres détails étant donnés dans la section 2.2.2.3.2.

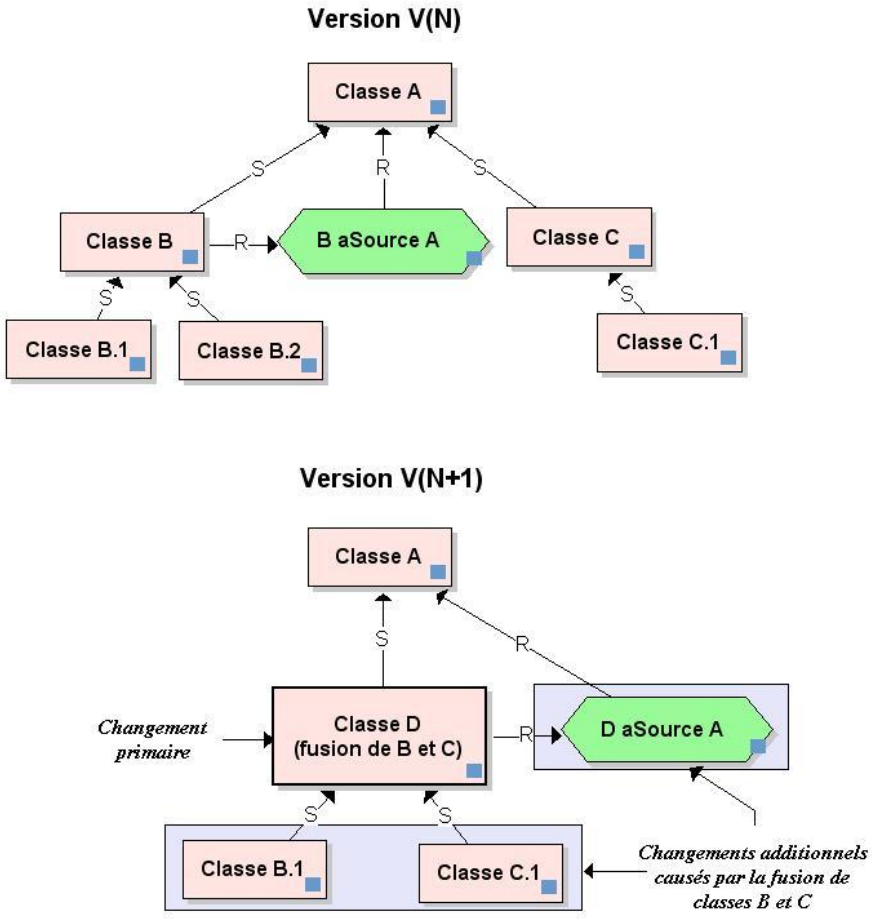


Figure II-3. Le changement primaire « Fusion de B et C » et ses changements additionnels de transfert de sous-classes et d'une propriété
(La légende du formalisme graphique MOT+Onto est consultable dans l'Appendice C)

2.2.2. **Méthodologie d'évolution des ontologies**

Nous avons conçu la méthodologie d'évolution comme un processus ayant au centre l'utilisateur, défini comme tout ontologiste responsable, à un moment donné, de l'évolution d'une ontologie (Rogozan, 2004a; Rogozan, Paquette, et Rosca, 2004). Ce processus est illustré dans la Figure II-4 et ses étapes sont discutées brièvement ci-après. Précisons que chaque étape devrait être supportée informatiquement par des outils spécialisés comme des éditeurs d'ontologies, des modules de journalisation et d'analyse des changements ou encore des systèmes permettant la mise en opération de la nouvelle version de l'ontologie.

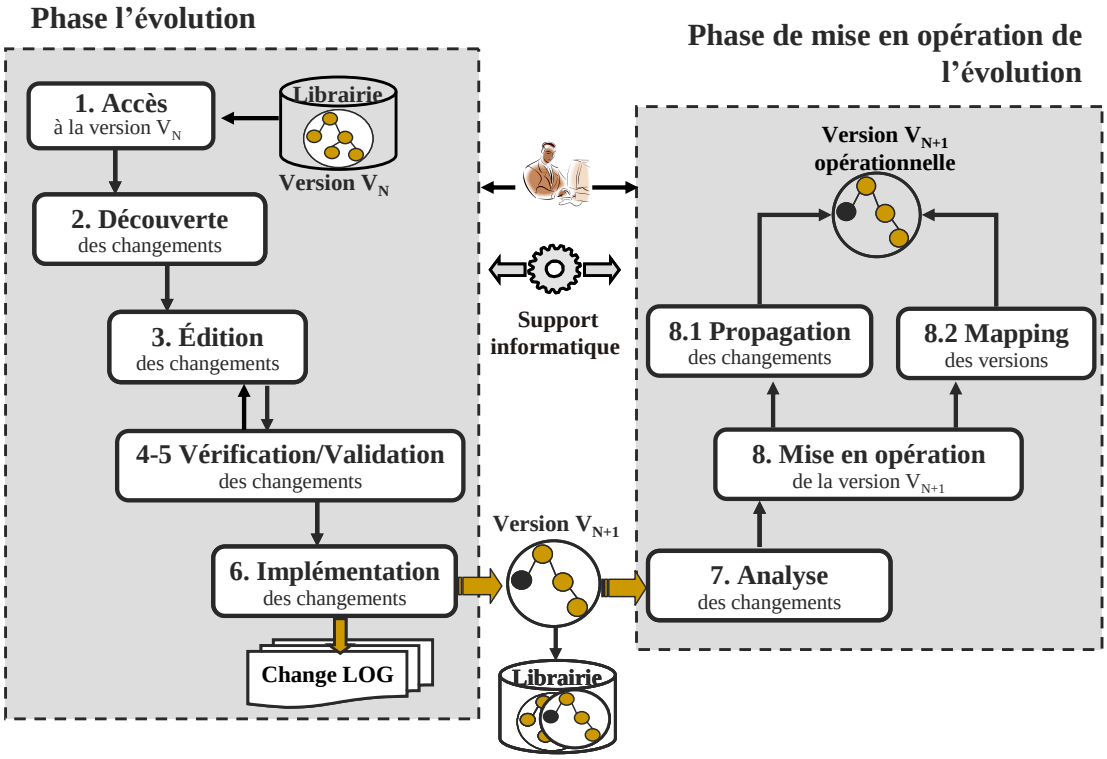


Figure II-4. Modèle de la méthodologie d'évolution d'ontologies

2.2.2.1. **Étape 1. Accéder, identifier et télécharger la version de l'ontologie à modifier**

Les ontologies se trouvent généralement stockées dans des librairies distribuées sur le web qui fournissent des mécanismes d'emmagasinage, permettant un accès constant aux ontologies ainsi que la possibilité de naviguer dans les ontologies (Ding et Fensel, 2001). À cette étape, les utilisateurs doivent accéder à l'ontologie à modifier – la **version V_N** – et la télécharger sur leur poste de travail.

2.2.2.2. **Étape 2. Découvrir les changements**

Les utilisateurs découvrent les changements à apporter à la version V_N en fonction de : (a) modifications apparues dans le domaine de définition ou d'utilisation de l'ontologie; (b) une analyse de l'ontologie par d'heuristiques, comme celle affirmant que, si une classe

possède une seule sous-classe, les deux peuvent être fusionnées ou encore, si toutes les sous-classes ont une même propriété, alors cette propriété peut être déplacée à la classe-parent (Stojanovic, 2004).

2.2.2.3. **Étape 3 : Éditer les changements**

2.2.2.3.1. *Changements élémentaires et complexes*

À cette étape, les utilisateurs éditent les changements identifiés précédemment. En ce qui concerne les fonctionnalités pour permettre l'édition des changements, Klein, (2004), propose une taxonomie des changements élémentaires spécifiant l'ajout, l'effacement et la modification des classes et propriétés définies à l'aide de OWL-LITE. Nous avons prolongé cette taxonomie à OWL-DL et nous avons développé une taxonomie des changements complexes précisant, par exemple, le déplacement, la fusion ou la séparation des classes et/ou propriétés d'une ontologie OWL-DL. Nous avons regroupé ensuite ces deux taxonomies dans une ontologie de changements et nous avons introduit un certain nombre de caractéristiques afin de mieux spécifier le type et l'effet de chaque changement sur l'ontologie. L'ontologie de changements est discutée en détail dans le chapitre III.

2.2.2.3.2. *Changements primaires et additionnels*

Un changement dans l'ontologie peut demander d'autres changements additionnels, chacun d'entre eux conduisant à un état final consistant (Stojanovic, Maedche, Motik, et Stojanovic, 2002). Par exemple, si une classe à l'intérieur de la hiérarchie des classes est supprimée, ses sous-classes peuvent être soit supprimées, soit rattachées à la classe-parent de la classe supprimée, soit rattachées à la classe-racine de la hiérarchie. Ou encore, si deux classes sont fusionnées (comme dans la Figure II-3), leurs sous-classes et propriétés peuvent être effacées ou déplacées à la classe résultante ou à une autre. Dans la Figure II-5, nous présentons alors des exemples de stratégies d'évolution pour un **changement** de fusion de classes. Par **stratégie d'évolution,** nous comprenons la manière dont les utilisateurs décident de résoudre les inconsistances issues de l'application d'un changement primaire, par l'application d'un nombre des changements additionnels.

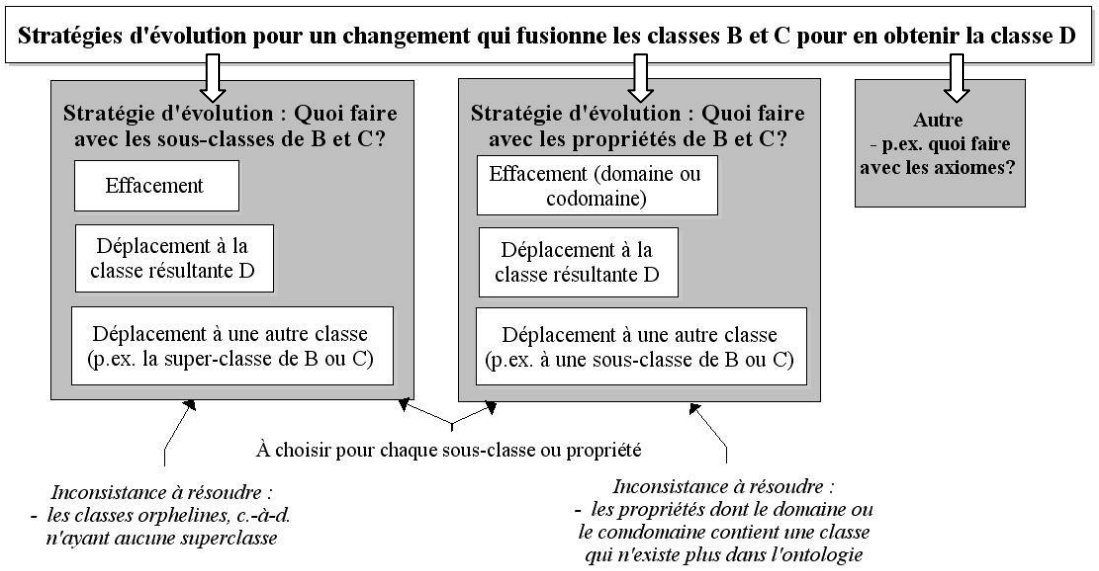


Figure II-5. Exemples de stratégies d'évolution pour un changement de fusion de classes

Le résultat de l'étape d'édition consiste alors dans une séquence des changements, chaque changement ayant la forme d'un couple (changement édité, stratégie de changements additionnels). De plus, chaque changement devrait être annoté en utilisant un ensemble de métadonnées décrivant l'auteur, le but et le rôle sémantique de l'entité ontologique étant le sujet du changement (Klein, Kiryakov, Ognyanov, et Fensel, 2002; Oliver, Shahar, Musen, et Shortliffe, 1999).

2.2.2.4. **Étape 4. Vérifier les changements**

Cette étape vise la vérification des effets des changements sur la consistance de l'ontologie. S'il reste encore des changements qu'invalident les contraintes de l'ontologie, alors les auteurs de l'évolution doivent être informés et une méthode pour les supporter dans la résolution d'inconsistances par l'ajout des changements additionnels doit leur être fournie (retour à l'étape d'édition de changements).

2.2.2.5. **Étape 5. Valider les changements**

Dans cette étape, la séquence des changements est validée conjointement par les auteurs de l'évolution de l'ontologie. Si cette validation n'est pas possible en raison du caractère distribué de l'environnement d'occurrence du processus d'évolution, alors des mécanismes de gestion des conflits entre des changements doivent être prévus (Pinto, Staab, et Tempich, 2004; Sunagawa, Mizoguchi, et all., 2003).

2.2.2.6. **Étape 6. Implémenter les changements**

Cette étape vise l'implémentation des changements avec une préservation de la trace[17] des changements apportés et des métadonnées qui leur sont associées. Elle vise également une gestion des identifiants des versions, sachant qu'on obtient maintenant une nouvelle version de l'ontologie, la **version V_{N+1},** avec un nouvel identifiant (un *URI*). Au besoin, les auteurs de l'évolution peuvent retourner maintenant à l'étape 2 du processus, l'ontologie à modifier étant maintenant la nouvelle version obtenue à la fin de l'étape 6.

2.2.2.7. **Étape 7. Analyser les changements**

Le processus d'évolution peut causer l'altération des rôles de l'ontologie (*cf.* Figure II-6). Par exemple, les systèmes conçus sur la base d'ontologies évolutives pourraient présenter des dysfonctionnements après l'application d'un certain nombre de changements, particulièrement si l'ontologie est au cœur de l'architecture fonctionnelle comme c'est le cas du système TELOS (Paquette, Rosca, Mihaila, et Masmoudi, 2007). Ou encore les ontologies dépendantes, c'est-à-dire celles qui incluent, dans leur structure ontologique, une partie de l'ontologie évolutive, peuvent présenter des inconsistances après l'évolution de cette ontologie (Maedche, Motik, et Stojanovic, 2003). Par exemple, l'effacement d'une classe de V_N qui est la racine d'une hiérarchie de classes dans une ontologie dépendante.

[17] La plupart du temps, cette trace prend la forme d'un journal de changements.

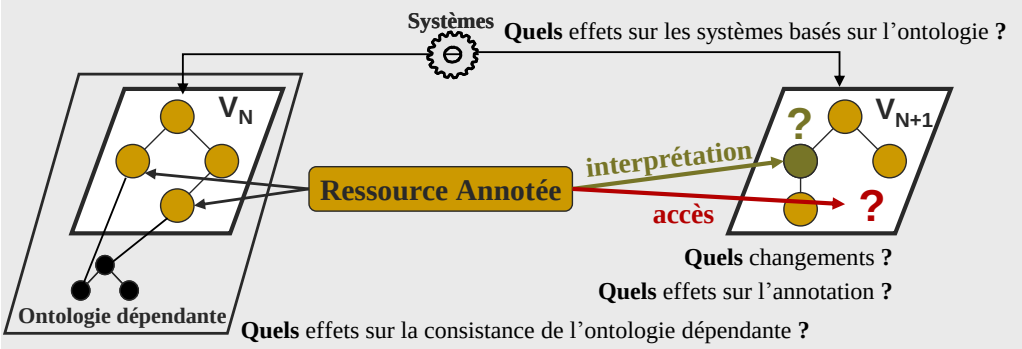


Figure II-6. Altération des rôles de l'ontologie suite à l'évolution

Finalement, considérons une ressource[18] référencée sémantiquement à l'aide d'une classe appartenant à une version V_N. Si cette classe est effacée en V_{N+1}, l'accès à la ressource est brisé ou, si la classe est modifiée, l'interprétation de la ressource, acquise au moyen de son référencement, peut manquer de cohérence.

Par conséquent, pour assurer l'évolution des ontologies avec une préservation des rôles et, donc, de l'utilité même des ontologies en question, l'objectif de cette étape est de fournir une analyse des effets de changements afin d'identifier ceux susceptibles de provoquer des problèmes de dysfonctionnement de systèmes, des inconsistances d'ontologies dépendantes ou encore des pertes d'accès aux ressources référencées sémantiquement.

2.2.2.8. **Étape 8. Mettre en opération la V_{N+1}**

En fonction des résultats de l'analyse de changements, le problème le plus difficile est maintenant de préserver les rôles de l'ontologie pour la nouvelle version V_{N+1}. La préservation des rôles peut se faire selon deux types de situation :

1. Quand il est possible de modifier ou de mettre à jour les artefacts dépendants (c.-à-d. le référencement sémantique de ressources, les ontologies dépendantes et les systèmes orientés ontologie) en fonction des changements exécutés pour passer de V_N à V_{N+1}, alors on peut utiliser uniquement la nouvelle version de l'ontologie.

[18] Rappelons que le terme ressource dénote tout type d'objet identifiable, comme un document, un outil ou un service, une personne.

Nous empruntons à Stojanovic, Stojanovic et Handschuh, (2002), le terme « propagation des changements » pour désigner cette situation.

2. Quand il n'est pas possible de modifier ou mettre à jour les artefacts dépendants en fonction de l'évolution de V_N à V_{N+1}, il faudrait spécifier, en se basant sur des approches de *mapping* inter-ontologies (Wache et *al.*, 2001), des liens de correspondance sémantique, comme l'équivalence, l'intersection ou l'incompatibilité, entre les classes et les propriétés appartenant à ces deux versions. Ceci permettrait alors l'utilisation des versions multiples pour assurer une préservation des rôles d'une ontologie.

2.2.3. **Mise en perspective de la méthodologie**

En définissant cette ébauche de méthodologie, nous avons identifié et décrit les étapes essentielles à prendre en compte pour supporter la gestion de l'évolution des ontologies dans un environnement dynamique, multi-acteurs et distribué. Nous sommes consciente que la réalisation de chaque étape demande une méthode spécifique, chaque méthode pouvant être considérée comme un sujet à débattre lors d'un projet de doctorat. Dans cette thèse, nous n'avons pas finalement l'ambition de développer des méthodes nouvelles ou d'adapter celles existantes pour permettre l'accomplissement de chacune des étapes. Nous n'avons donc que partiellement atteint notre premier objectif de recherche, mais suffisamment pour que l'on puisse analyser les outils existants en fonction du cadre conceptuel fourni par cette ébauche de méthodologie. Ainsi, dans le paragraphe suivant, nous analysons les fonctionnalités fournies actuellement par des outils de construction d'ontologie par rapport à chaque étape du processus d'évolution, le but étant d'identifier les besoins technologiques ressentis et de proposer des solutions pour les combler.

2.3 **Quels outils pour supporter l'évolution de l'ontologie?**

De nombreux outils permettent de construire de nouvelles ontologies ou de réutiliser ou modifier des ontologies existantes (OntoWeb, 2002b). Dans cette section, nous allons premièrement analyser l'existant en nous orientant vers les outils qui sont plus connus à

l'heure actuelle. Deuxièmement, tout en nous positionnant par rapport aux conclusions de l'analyse, nous proposons deux modules informatiques interconnectés qui forment ensemble ce que l'on appelle un **cadre de référence pour la gestion des changements** apportés aux versions d'ontologie.

2.3.1. État de l'art des outils de support aux ontologies

Dans ce paragraphe, nous présentons six outils parmi les plus utilisés actuellement dans le contexte du Web sémantique (KAON, OntoEdit, Protege, OILEd, WebOde et HOZO) en les analysant en fonction d'un ensemble de critères que nous avons mis en évidence dans la méthodologie d'évolution des ontologies.

2.3.1.1. Présentation des outils

Le logiciel intégré **KAON**[19], dont l'architecture est fondée sur la méthodologie de l'AIFB (*Institute of Applied Informatics and Formal Description Methods*), est le seul ayant été spécifiquement conçu pour permettre la construction et l'évolution des ontologies (Oberle, Volz, Motik, et Staab, 2004). Il offre une suite d'outils pour supporter l'édition des changements, mais aussi la vérification de la consistance des ontologies couplée avec la possibilité de créer des stratégies d'évolution personnalisées. Pour ce qui est de la représentation formelle d'une ontologie, l'outil est encore basé sur un langage qui lui est propre – le langage KAON – même si des travaux récents indiquent un passage vers OWL.

OntoEdit[20], développé lui aussi dans le cadre de l'AIFB, s'appuie également sur une réflexion méthodologique significative, la méthodologie On-To-Knowledge (Sure, Staab, et Studer, 2004). Comme tous les autres éditeurs, il permet l'édition des hiérarchies de classes et de relations ainsi que l'édition d'axiomes. Des outils graphiques dédiés à la visualisation de l'ontologie sont inclus dans l'environnement ainsi qu'un contrôle de la cohérence de l'ontologie par une gestion des ordres d'édition. C'est principalement un environnement de

[19] KAON (KArlsruhe ONtology and Semantic Web infrastructure): http://kaon.semanticweb.org/.
[20] L'outil n'est pas disponible gratuitement dans sa version complète : une version de démonstration est disponible sur le site d'Ontoprise, à l'adresse http://www.ontoprise.com/.

construction d'ontologies permettant l'import et l'export des ontologies vers de multiples langages.

Un autre outil qui a comme base un support méthodologique, **WebODE**[21], assure presque fidèlement les critères de la méthodologie METHONTOLOGY développée par Arpirez, Corcho, Fernandez-Lopez et Gomez-Perez (2001). C'est une plateforme de conception d'ontologie, fonctionnant en ligne, où l'accent est mis sur la possibilité d'un travail collaboratif ainsi que sur la mise à disposition d'outils complémentaires, comme un moteur d'inférences et d'une interface graphique bien travaillée. WebOde fournit aussi des traducteurs permettant d'importer et d'exporter des ontologies en divers langages, comme le DAML+OIL, l'OIL, le RDF(S) ou l'XML.

OILEd[22] (Bechhofer et Goble, 2001) a été développé par l'université de Manchester. Il est dédié essentiellement à la construction de petites ontologies dans le langage de représentation OIL (*Ontology Inference Layer*) (Fensel et al., 2001), un des précurseurs du langage OWL (*Web Ontology Language*). Initialement conçu dans le but de démontrer les vertus du langage OIL, il offre néanmoins tous les éléments d'interface permettant de créer des hiérarchies de classes, d'axiomes ou de propriétés ainsi que les services d'un raisonneur, FaCT, pour tester la consistance des ontologies construites.

HOZO[23] a été conçu par le MizLab de l'université d'Osaka. À travers ses modules (OntologyEditor et OntoManager), il fournit aux utilisateurs une interface graphique pour éditer des ontologies nouvelles, mais aussi pour modifier celles existantes. L'outil permet également à chaque utilisateur de parcourir les ontologies des autres et, en suivant un modèle de résolution de conflits (Sunagawa, Mizoguchi, et all., 2003), d'accepter ou de rejeter les changements qui peuvent influencer la consistance de leur ontologie. Ces ontologies concurrentes sont disponibles dans plusieurs formats, comme le Lisp ou l'XML.

Finalement, **PROTÉGÉ** (Noy, Fergerson, et Musen, 2000), développé par l'équipe SMI (*Stanford Medical Informatics*) de l'université de Stanford, est une plateforme *open-*

[21] WebODE est accessible en ligne, à l'adresse : http://webode.dia.fi.upm.es/webode/login.html.
[22] OilEd est accessible en ligne, à l'adresse : http://oiled.man.ac.uk/.
[23] HOZO est accessible en ligne, à l'adresse : http://www.ei.sanken.osaka-u.ac.jp/english/.

source permettant la visualisation et la construction des ontologies ainsi que la vérification des contraintes reliées à la consistance. Il comprend principalement un éditeur à base de formulaire ainsi qu'une architecture modulaire permettant l'ajout de nouvelles fonctionnalités ou d'outils personnalisés. Le modèle de connaissances sous-jacent est issu du modèle des *frames* et contient des *classes* (concepts), des *slots* (attributs) et des *facets* (valeurs des attributs et contraintes). Un plugiciel pour le langage OWL[24] a été implémenté récemment à PROTÉGÉ afin de permettre l'édition des ontologies interprétables à travers le Web sémantique (Horridge. M., Knublauch, Rector, Stevens, et Wroe, 2004). D'autres plugiciels lui ont été associés, comme le PromptDiff (Noy et Musen, 2000) pour une mise en correspondance des ontologies ou comme le raisonneur RACER pour une vérification des incohérences dans l'ontologie. L'interface très complète ainsi que l'architecture logicielle ouverte ont grandement participé au succès de Protégé, celui-ci étant actuellement une des références dans le domaine de l'ingénierie ontologique.

2.3.1.2. **Analyse des outils**

Nous analysons maintenant ces outils selon les fonctionnalités nécessaires pour supporter les huit étapes du processus d'évolution. Pour alléger la présentation de l'analyse, nous avons regroupé ces fonctionnalités selon qu'elles correspondent à la phase d'évolution de l'ontologie (le Tableau II-1) ou à la phase de mise en opération de l'évolution (le Tableau II-2).

[24] Le Protégé-OWL est accessible en ligne, à l'adresse : http://protege.stanford.edu/overview/protege-owl.html.

Tableau II-1. Analyse des outils pour la phase d'évolution (étapes 1-6)

Légende : (Oui) - support complet, (S/P) – support partiel, (Non) - aucun support

Étape	Kaon	OntoE.	Protégé	OILEd	WebOde	Hozo
1	**Librairies d'ontologies** : l'outil offre la possibilité d'identification et d'accès aux ontologies stockées dans des libraires distribuées ainsi que l'archivage de l'ontologie (ou de la version) évoluée.					
	S/P	S/P	S/P	S/P	S/P	S/P
2	**Découverte de changements** : en utilisant des heuristiques, l'outil supporte les utilisateurs dans l'analyse de la structure de l'ontologie et dans la découverte de changements.					
	Non	Non	Non	Non	Non	Non
3	**Édition de changements élémentaires** : l'outil permet l'édition de changements élémentaires.					
	Oui	Oui	Oui	Oui	Oui	Oui
	Édition de changements complexes : l'outil permet l'édition de changements complexes.					
	S/P	S/P	Non	Non	Non	Non
	Stratégies d'évolution : pour chaque type de changement qui demande un certain nombre de changements additionnels, l'outil offre le choix parmi plusieurs stratégies d'évolution.					
	S/P	Non	Non	Non	Non	Non
4	**Vérification de la consistance :** l'outil vérifie que les changements n'ont pas généré des inconsistances concernant le modèle structural et axiomatique de l'ontologie (ou la version) évoluée.					
	Oui	Oui (*via* OntoBroker)	Oui (*via* PAL, FaCT)	Oui (*via* FaCT)	Oui (*via* OntoClean)	Oui
	Résolution d'inconsistances : l'outil permet la résolution des inconsistances générées par les changements (p.ex. par l'ajout des changements additionnels).					
	S/P	Non	Non	Non	Non	S/P (*via* OntoManager)
5	**Support à la résolution de conflits collectifs** : l'outil offre la possibilité de résoudre les conflits issus d'une édition ou d'une validation collective de la modification d'une ontologie					
	Non	Oui (*via* SWAP)	Oui (*via* ChAO)	Non	S/P	Oui (*via* OntoManager)
6	**Journalisation de changements** : l'outil enregistre dans un fichier-journal les changements exécutés lors de l'évolution de l'ontologie.					
	Oui	S/P	S/P	Non	Non	S/P

Tableau II-2. Analyse des outils pour la phase de mise en opération de l'évolution (étapes 7 et 8)

Légende : (Oui) - support complet, (S/P) – support partiel, (Non) - aucun support

Étape	Kaon	OntoE.	Protégé	OILEd	WebOde	Hozo
7	**Identification des changements appliqués** : l'outil permet l'identification de l'ensemble des changements apportés lors de l'évolution d'une ontologie.					
	S/P	Non	S/P (*via* PromptDIff)	Non	Non	Non
	Analyse de l'effet de changements sur le référencement sémantique : l'outil fournit une analyse des effets de changements sur l'accès et sur l'interprétation de ressources référencées sémantiquement.					
	Non	Non	Non	Non	Non	Non
	Analyse de l'effet de changements sur les ontologies dépendantes : l'outil fournit une analyse des effets de changements sur la consistance d'ontologies dépendantes et leur interprétation					
	S/P	Non	Non	Non	Non	S/P
8	**Propagation de changements dans le référencement sémantique** : l'outil offre un support pour la mise à jour des références sémantiques de ressources en fonction de changements apportés à l'ontologie.					
	Non	Non	Non	Non	Non	Non
	Propagation de changements dans les ontologies dépendantes : l'outil offre un support pour la résolution des inconsistances introduites par des changements apportés à l'ontologie source.					
	S/P	Non	Non	Non	Non	S/P
	Mise en correspondance de versions d'ontologie : l'outil est capable d'effectuer des mises en correspondances entre les versions d'une même ontologie					
	Non	Non	S/P (*via* AnchorPrompt)	Non	Non	Non

L'analyse des outils a mis en évidence plusieurs aspects, comme nous le montrons ci-après :

- **Étape 1** : Les outils fournissent uniquement des fonctionnalités pour accéder et pour stocker les ontologies sous la forme de fichiers, dans une base de données.
- **Étape 2** : Les outils n'offrent aucun support pour la découverte de changements.
- **Étape 3** : Tous les outils possèdent des fonctionnalités pour éditer les changements élémentaires, mais seulement KAON et OntoEdit offrent la possibilité d'éditer un

seul type de changement complexe : le déplacement des classes. Quant aux stratégies d'évolution, la plupart des outils offrent une seule possibilité pour résoudre les changements (p.ex. le changement `DeleteClass` efface aussi toutes les sous-classes de la classe effacée). Seulement KAON autorise les utilisateurs à choisir parmi un nombre de stratégies possibles et d'adapter ainsi la manière de résolution de certains changements. En ce qui concerne l'annotation des changements, les outils permettent seulement la description textuelle d'une entité ontologique, mais ils ne donnent pas la possibilité d'annoter les changements en utilisant un ensemble de métadonnées spécifiques.

- **Étape 4** : Tous les outils assurent la vérification de la consistance de l'ontologie soit au moyen d'un raisonneur interne à l'outil, soit en s'assurant les services d'un raisonneur comme FaCT (Horrocks, Sattler, et Tobies, 1999), qui est fondé sur la logique de description ou comme OntoBroker (Oberle, Volz, Motik, et Staab, 2004), basé sur la logique de *frames*, ou encore comme OntoClean (Guarino et Welty, 2002), qui permet l'évaluation des taxonomies de classes selon des notions comme rigidité, unité, identité. Cependant, seul KAON possède la capacité de résoudre un certain nombre des inconsistances trouvées, en proposant des stratégies d'évolution pour résoudre l'inconsistance due à une classe orpheline, par exemple. Quant à HOZO, il permet la résolution des inconsistances entre une ontologie partagée et des ontologies locales, qui en dépendent.

- **Étape 5** : KAON et WebOde permettent aux utilisateurs de modifier (ou plutôt de construire) la même ontologie en leur fournissant un mécanisme de synchronisation permettant la gestion des situations conflictuelles. Protége utilise un plugiciel, axé sur C_HAO (Noy, Chugh, Liu, et Musen, 2006), qui permet aux utilisateurs multiples d'accepter ou rejeter les changements qui leur sont présentés lors d'une comparaison des versions d'ontologie, d'ajouter leurs commentaires sur la décision qu'ils ont prise et de voir aussi les commentaires des autres. Pour ce qui est d'OntoEdit et HOZO, les deux implémentent un modèle d'édition collaborative fondé sur la négociation entre des acteurs lors de la modification des ontologies partagées : le modèle de Pinto,

Staab et Tempich, (2004), pour OntoEdit et le modèle de Sunagawa, Mizoguchi et *al.*, (2003), pour HOZO.

- **Étapes 6 et 7** : La plupart des outils possèdent un journal des changements, mais celui-ci est toujours exprimé dans un langage orienté outil, interprétable uniquement par l'outil en question. Seul KAON possède une fonction spécialement dédiée à la journalisation des changements qui enregistre dans un fichier-journal, sous une forme structurée et standardisée, les changements apportés lors de l'évolution d'une ontologie. Ainsi, en tout temps après l'évolution, KAON est capable d'effectuer une identification des changements élémentaires à partir de son journal. Quant à Protégé, il possède un plugiciel, le PromptDiff (Noy et Musen, 2003b), qui est capable d'identifier si des classes ou propriétés ont été modifiées ou non dans la nouvelle version de l'ontologie. Cependant, aucun des outils ne fournit un support pour l'analyse des effets des changements sur le référencement sémantique.

- **Étape 8** : Les outils ne proposent aucun support pour la propagation des changements dans les ressources référencées sémantiquement. Quant à la propagation dans les ontologies dépendantes, KAON fournit un support partiel puisqu'il ne prend pas en compte les changements complexes, mais seulement les changements élémentaires de type ajout ou effacement de classes et propriétés (Maedche, Motik, et Stojanovic, 2003; Stojanovic, 2004). Quant à HOZO, il considère la modification des parties dépendantes d'une ontologie en fonction de l'avis donné par les acteurs responsables de chacune de ces parties (Kozaki, Sunagawa, Kitamura, et Mizoguchi, 2007; Sunagawa, Mizoguchi, et all., 2003). Finalement, Protégé est le seul qui peut supporter la mise en correspondance entre deux versions de l'ontologie au moyen du plugiciel AnchorPROMPT (Noy et Musen, 2003b) qui est un outil graphique capable d'assister les utilisateurs dans l'identification des similarités entre différentes ontologies.

2.3.1.3. **Conclusion et présentation d'objectifs spécifiques de la recherche**

L'analyse des principaux outils a mis en évidence l'absence de fonctionnalités importantes pour supporter principalement l'évolution des ontologies. Les premières étapes

de l'évolution sont assez bien supportées par les outils, les exceptions étant la découverte de changements, l'édition de changements complexes ainsi que la journalisation significative et normalisée du processus d'évolution. Les deux dernières, concernant l'identification des changements, l'analyse de leurs effets ainsi que la propagation de changements dans le référencement sémantique des ressources, ne sont que très faiblement supportées par les outils cités ci-dessus. Pourtant, sans savoir comment les ontologies ont été modifiées, à la suite de quels types de changements, avec quels types d'effet, la préservation des rôles d'ontologies est difficilement réalisable, sinon impossible.

Dans cette thèse, nous nous donnons comme but de développer des solutions pour répondre à ces carences, des prototypes exploratoires de nouvelles idées sur la manière de journaliser les changements et de préserver le rôle des ontologies utilisées comme référentiel sémantique, le tout regroupé dans un **cadre de référence pour la gestion des changements apportés aux versions d'ontologies**. Nous énonçons alors nos objectifs de recherche sur le plan technologique qui visent :

1. La conception d'une technique, implémentée dans le *ChangeHistoryBuilder* (CHB), pour tracer et identifier les changements apportés aux ontologies du web sémantique ainsi que la conception d'une ontologie des changements. Nous présentons les résultats reliés à cet objectif dans les chapitres III et IV.

2. La mise en évidence des effets de changements sur le référencement sémantique ainsi que le développement des solutions pour maintenir l'accès et l'interprétation de ressources référencées après l'évolution des ontologies. Par **ressource référencée sémantiquement** nous comprenons tout type d'objet auquel on associe des références sémantiques (p.ex. des classes) provenant d'une ou plusieurs ontologies. Dans le Chapitre V, nous présentons les résultats reliés à cet objectif et le système SemanticAnnotationModifier (SAM) qui les implémente.

2.3.2. **Vers des outils de support à la gestion des changements**

La gestion des changements apportés aux versions d'ontologie comprend plusieurs aspects majeurs. Le premier concerne l'édition et la vérification de la consistance de

changements durant le processus d'évolution. Habituellement effectué par les éditeurs d'ontologies, cet aspect touche à l'exécution des changements d'une manière adéquate. Le deuxième aspect vise le traçage ou l'identification des changements après l'évolution ainsi que l'analyse des effets de changements sur les artefacts dépendants (p.ex. ressources référencées sémantiquement, ontologies dépendantes). Enfin, le troisième aspect touche à la résolution des effets de changements sur les artefacts dépendants (p.ex. la maintenance de l'intégrité du référencement sémantique après l'évolution ou la maintenance de l'interopérabilité avec les ontologies dépendantes).

Dans le cadre de référence présenté dans cette thèse, on peut retrouver tous ces aspects de la gestion des changements. Précisons cependant les deux derniers aspects sont ciblés sur le référencement sémantique des ressources. Ainsi, dans la section 2.3.2.1. nous présentons les principes qui nous ont orientés dans la conception du cadre de gestion de changements et dans la section 2.3.2.2. nous illustrons brièvement les modules informatiques qui le composent.

2.3.2.1. **Principes d'orientation du cadre de référence pour la gestion de changements**

La conception du cadre de référence pour la gestion de changements apportés aux versions d'ontologie est fondée sur un certain nombre de principes :

- Réduire les risques inhérents à l'évolution technologique rapide dans le contexte du Web sémantique, en mettant l'accent sur des analyses et des évaluations préliminaires et en laissant le développement plus coûteux des systèmes pour plus tard. Le principe est de développer des **prototypes exploratoires** pour mettre en évidence les solutions conceptuelles que nous proposons pour la gestion des changements ontologiques.

- Rendre modulaire le cadre de référence en développant des composants à granularité moyenne – des **modules logiciels** – à être utilisés indépendamment ou en combinaison.

- Rendre les solutions conceptuelles et les fonctionnalités logicielles qui en découlent indépendantes d'un environnement technologique précis. Le principe ici est d'assurer la généricité et la réutilisabilité des modules qui composent le cadre de référence, en le concevant comme des **plugiciels** qui peuvent être associés à divers éditeurs d'ontologies.

- Voir le cadre de gestions de changements comme une **société d'agents humains et informatiques** interagissant ensemble. Il arrive que faire appel à l'intervention humaine pour décider de la cohérence des services offerts puisse être plus adéquat que la construction des systèmes complètement automatisés. C'est le cas de la gestion de l'évolution des ontologies et de son impact sur l'interprétation des ressources référencées sémantiquement.

2.3.2.2. **Cadre de référence pour la gestion de changements**

Dans la Figure II-7, nous présentons brièvement le cadre de référence en mettant en évidence les deux modules que nous proposons dans cette thèse – (1) *ChangeHistoryBuilder* (**CHB**) ; (2) *SemancticAnnotationModifier* (**SAM**) – et leurs interactions respectives.

Éditeur d'ontologies

Un éditeur d'ontologies peut être utilisé soit pour construire une nouvelle ontologie, soit pour réutiliser des ontologies existantes et les modifier. Il offre des fonctions communes d'édition de changements élémentaires et/ou complexes, de vérification de la consistance de changements, d'enregistrement des changements dans un journal et d'exportation des ontologies dans un des langages de représentation appropriée, tels que le langage OWL.

ChangeHistoryBuilder (CHB)

Le module CHB possède deux fonctionnalités de base. La première est celle de journaliser les changements, lors de la phase d'évolution de l'ontologie. Pour communiquer avec divers éditeurs d'ontologies lors de la journalisation, le CHB leur procure un modèle de métadonnées que ces éditeurs peuvent utiliser pour capter les changements d'une manière standardisée et riche sémantiquement. La deuxième fonctionnalité est celle de récupérer la trace de changements, d'extraire les changements et de les présenter à l'utilisateur. Cette

présentation peut être accompagnée par une analyse des effets de changements sur le référencement sémantique de ressources, fournie par le module SAM, avec lequel le CHB peut communiquer.

SemanticAnnotationModifier (SAM)

Le module SAM part de l'idée que, pour garantir une bonne évolution de l'ontologie, il faudrait maintenir aussi l'intégrité de ce qui repose déjà sur l'ontologie, en plus de l'ontologie elle-même. En effet, lorsque l'ontologie change, ses changements ont un impact sur tout ce qui a été construit au-dessus, implicitement sur les banques de ressources référencées sémantiquement. C'est dans ce contexte que SAM trouve son entière utilité, en offrant aux utilisateurs: (1) un service d'information permettant de visualiser et de comprendre si un certain changement engendre une perte d'accès aux ressources référencées ou une modification de l'interprétation de ressources ; (2) un service de type conseils pour guider les utilisateurs lors du processus de mise à jour du référencement qui permet à toutes les ressources d'être accessibles et correctement décrites via la nouvelle version de l'ontologie évoluée.

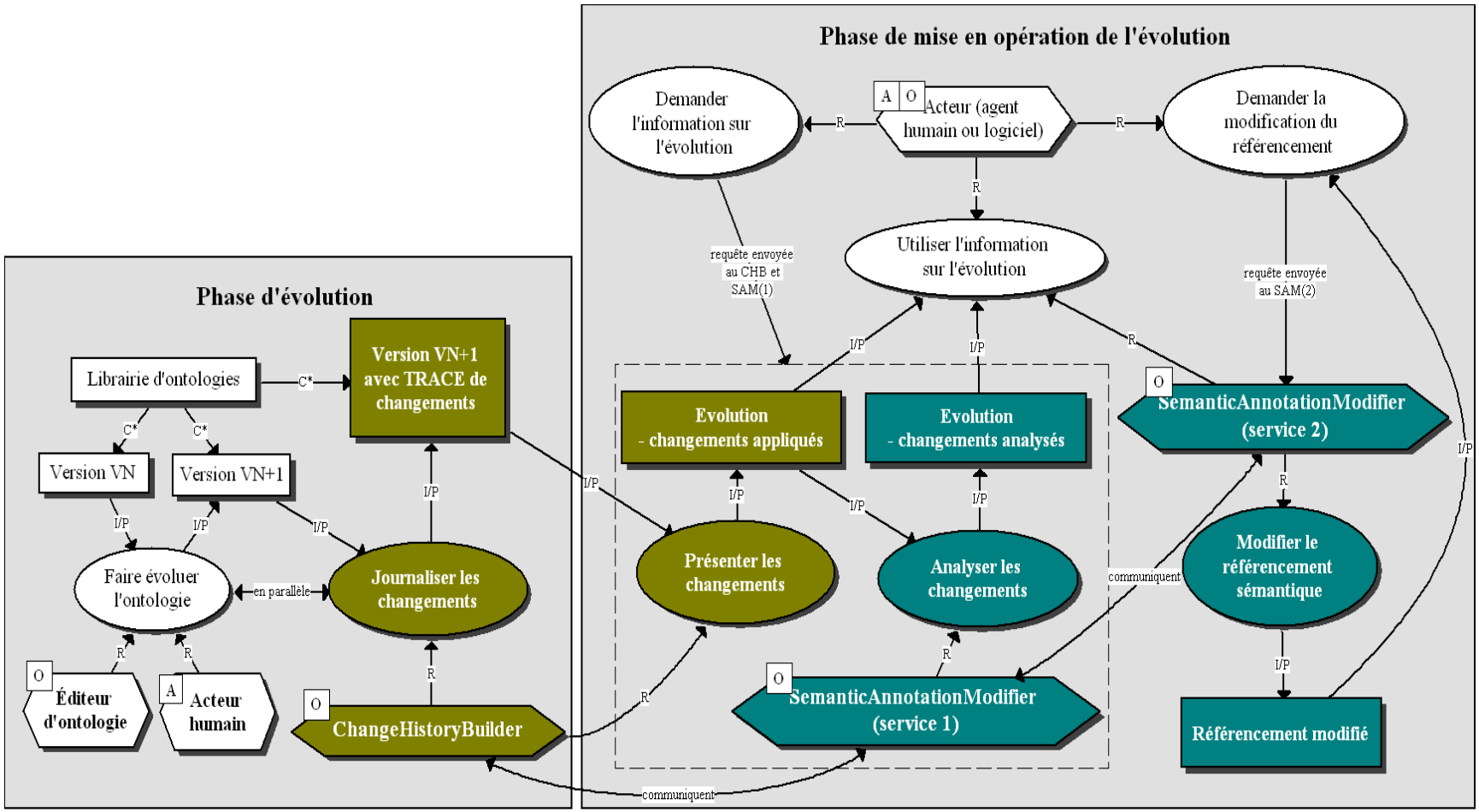


Figure II-7 Cadre de référence pour la gestion de changements apportés aux versions d'ontologie
(La légende du formalisme graphique MOT est consultable dans l'Appendice A)

2.4 **Conclusion**

Ce chapitre correspond au premier objectif de notre thèse, à savoir : concevoir une méthodologie présentant une vue unifiée sur l'évolution de l'ontologie. Pour y répondre, nous avons d'abord présenté un état de l'art des approches méthodologiques actuelles. Nous avons ensuite proposé un cadre intégrateur qui les unifie dans un processus complet d'évolution des ontologies. Ce processus se compose de huit étapes et possède deux phases : une phase d'évolution et une phase de mise en opération de l'évolution d'une ontologie.

Pour identifier ce qui existe actuellement en termes d'outils de support à l'évolution des ontologies, nous avons analysé les fonctionnalités offertes par divers éditeurs d'ontologies en fonction de chacune des huit étapes du processus d'évolution proposé. Nous sommes ainsi arrivée à identifier des carences importantes dans la gestion de changements d'ontologies, particulièrement, en ce qui concerne la journalisation et l'identification de changements élémentaires et complexes, l'analyse des effets de changements sur le référencement sémantique des ressources ainsi que le maintien de l'intégrité de ce référencement. Nous nous sommes fixé ensuite comme objectifs de combler ces carences en proposant un cadre de référence pour la gestion des changements apportés à une version d'ontologie (V_N) pour obtenir une nouvelle version d'ontologie (V_{N+1}). Ce cadre se compose d'une ontologie des changements (*cf.* Chapitre III), du système *ChangeHistoryBuilder* (*cf.* Chapitre IV) et du système *SemanticAnnotationModifier* (*cf.* Chapitre V).

Chapitre III

ONTOLOGIE DES CHANGEMENTS

Dans le chapitre précédent, nous avons présenté notre ébauche d'une méthodologie d'évolution des ontologies et nous avons identifié des besoins technologiques pour la soutenir. Pour combler ces besoins, nous nous sommes donné comme objectif de concevoir un cadre de référence pour la gestion des changements apportés aux versions d'ontologie du web sémantique. Dans ce contexte, savoir ce que sont ces changements, leur type, leur degré de complexité, leur signification précise devient une question essentielle. Ainsi est née l'idée d'une ontologie des changements.

Dans ce chapitre, nous présentons l'ontologie des changements que nous avons conçue pour décrire explicitement l'ensemble des changements, élémentaires et complexes, applicables à des ontologies OWL (les ontologies du Web sémantique). Nous commençons par faire l'état de l'art des classifications existantes et par discuter leurs apports et leurs limites (*cf.* Section 3.1). Nous présentons ensuite notre ontologie de changements, qui est, à notre connaissance, la plus complète actuellement et la seule qui offre une spécification des changements complexes et de leurs propriétés (*cf.* Section 3.2). Le langage de représentation choisi est le formalisme standard OWL. Nous obtenons ainsi une ontologie véritablement partageable et exploitable par tout agent conforme OWL. Nous finissons le chapitre par les conclusions (*cf.* Section 3.3).

3.1 **Classification des changements : état de l'art**

3.1.1. **Classifications des changements proposées dans la documentation du domaine**

Pour caractériser les modifications apportées à une ontologie, Stojanovic, (2004) introduit trois niveaux de classification (*cf.* Figure III-1) : les changements élémentaires, composites et complexes. Un **changement élémentaire** est un changement primitif, qui ajoute ou retire un seul élément de l'ontologie. Un **changement composite** est celui qui crée, retire ou modifie un et seulement un niveau du voisinage d'un élément de l'ontologie. Un exemple d'un tel changement est « *Remonter un concept* » (annexer une classe C à toutes les classes-parent de ses parents) ou « *Généraliser un concept* » (ajouter une nouvelle classe entre une classe C et tous ses parents). Enfin, un **changement complexe** est un changement qui peut être décomposé dans n'importe quelle combinaison d'au moins deux changements élémentaires ou composites. Un exemple d'un tel changement est « *Déplacer un concept* » qui est la généralisation de « *Remonter un concept* » (ou « *Descendre un concept* ») vu qu'il attache une classe à une autre qui ne fait pas nécessairement partie de son voisinage.

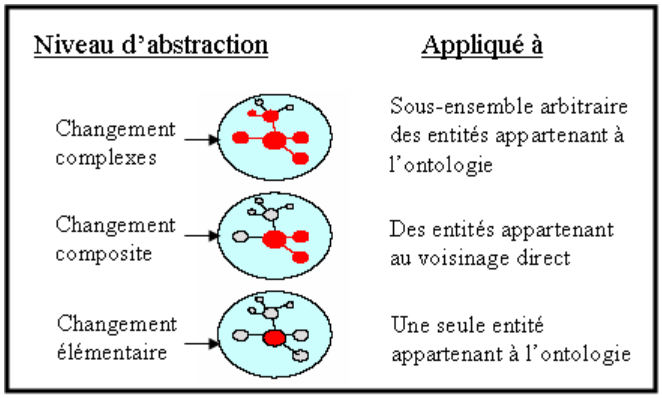


Figure III-1. Caractérisation de changements en fonction de leur niveau d'abstraction, selon Stojanovic, (2004)

Cependant, même si l'auteur introduit la notion de classification de changements par niveaux d'abstraction, il ne développe ensuite que le niveau le plus bas. En appliquant les méta-changements « *Ajouter* » et « *Retirer* » à chaque élément d'une ontologie, l'auteur

propose une taxonomie de changements élémentaires (*cf.* Tableau III-1) applicables à des ontologies KAON (Oberle, Volz, Motik, et Staab, 2004).

Tableau III-1 Taxonomie des changements élémentaires selon Stojanovic, (2004)

Élément d'une ontologie KAON	Add	Remove
Concept	AddConcept	RemoveConcept
Concept Hierarchy	AddSubConcept	RemoveSubConcept
Property	AddProperty	RemoveProperty
Property Hierarchy	AddSubProperty	RemoveSubProperty
Property Domain	AddPropertyDomain	RemovePropertyDomain
Property Range	AddPropertyRange	RemovePropertyRange
Property Symmetric	AddPropertySymmetric	RemovePropertySymmet.
Property Transitive	AddPropertyTransitive	RemovePropertyTransit.
Property Inverse	AddPropertyInverse	RemovePropertyInverse
Max Cardinality	AddMaxCardinality	RemoveMaxCardinality
Min Cardinality	AddMinCardinality	RemoveMinCardinality

Klein (2004), présente lui aussi un ensemble de changements, mais il regarde plus du côté des ontologies OWL (*cf.* Tableau III-2). En comparant ses résultats avec la taxonomie mentionnée ci-dessus, cet auteur introduit en plus les opérations suivantes : (a) l'opération de sélection ou désélection de caractéristiques de propriétés comme la transitivité ou la symétrie ; (b) des opérations d'ajout et d'effacement d'une relation de sous-classe entre deux classes ou d'une relation de sous-propriété entre deux propriétés ; (c) des opérations d'ajout et d'effacement d'une d'équivalence ou d'une disjonction entre deux classes.

Tableau III-2 Taxonomie des changements élémentaires selon Klein, (2004)

Élément d'une ontologie OWL	Opération de changement
Opérations appliquées aux classes	
class	add, remove
cardinality upper/lowerbound	add, remove
cardinality upper/lowerbound	modify
superclass relation	add, remove
class disjointness	add, remove
class equivalence	add, remove
Opérations appliquées aux propriétés	
property	add, remove
domain	add, remove
range	add, remove
superproperty relation	add, remove
property equivalence	add, remove,
property inverse	add, remove
property symmetry	set, unset
property functionality	set, unset
property transitivity	set, unset
property inverse-functionality	set, unset

L'auteur introduit aussi une caractérisation des changements en fonction de deux dimensions : atomique *vs* composite, simple *vs* riche (*cf.* Tableau III-3).

Tableau III-3 Caractérisation bidimensionnelle de changements, selon Klein, (2004)

Changement	**simple**	**riche**
atomique	élémentaire	complexe
composite	complexe	complexe

Les changements atomiques sont des opérations indivisibles, alors que ceux composites sont des opérations combinées de changements atomiques. Les changements simples sont ceux qui peuvent être identifiés en analysant uniquement la structure de l'ontologie. L'identification de changements riches requiert une interrogation de la théorie logique de l'ontologie, étant donné que ces changements englobent une information à propos de ses conséquences sur le modèle logique de l'ontologie. Par exemple, un changement riche peut spécifier que le domaine d'une propriété est élargi puisque la classe appartenant au domaine a été remplacée par une de ses superclasses.

3.1.2. **Discussion des classifications proposées dans la littérature**

3.1.2.1. **Clarification de la terminologie**

Observons que les changements élémentaires et complexes, selon Stojanovic, (2004), correspondent à ceux atomiques et composites, selon Klein, (2004). Bien que le premier auteur ne fasse aucune distinction entre le caractère simple ou riche d'un changement, il fait une différenciation entre un changement complexe et composite, en introduisant ce dernier comme un intermédiaire entre les deux autres. Il nous semble cependant que cette différenciation peut laisser place à l'interprétation, étant donné que la limite entre un changement complexe et composite n'est pas nécessairement très précise. Par exemple, un changement complexe de déplacement de classe peut être, en même temps un composite, si le déplacement remonte (ou descend) la classe d'un seul niveau dans la hiérarchie. De ce point de vue, on trouve plus claire la seule distinction entre un changement élémentaire, qui est un changement primitif et non décomposable) et un changement complexe/composite, qui est

une combinaison de plusieurs changements élémentaires formant ensemble une seule entité logique.

3.1.2.2. **Changements complexes versus élémentaires**

Nous observons ensuite que les auteurs mentionnés ci-dessus, même s'ils définissent ce que sont les changements complexes (ou composites), ne les traitent pas d'une manière aussi poussée que les changements élémentaires. Bien qu'il soit vrai que les changements élémentaires permettraient l'édition de tout type de changement complexe, il est aussi vrai qu'il est beaucoup plus avantageux d'utiliser directement des changements complexes (Klein et Noy, 2003).

Avantages des changements complexes du point de vue de l'édition de changements

- Les changements complexes permettent aux acteurs humains de formuler les modifications de l'ontologie à un plus haut niveau d'abstraction, correspondant à leur modèle mental. Par exemple, pour déplacer une classe dans la hiérarchie des concepts, le changement complexe `MoveClass` possède un degré d'expressivité supérieur à une séquence de changements élémentaires `Delete` et `AddClass`.

- Les changements complexes facilitent la validation, collective ou non, de l'évolution d'une ontologie. Visualiser une seule opération de changement complexe (p.ex. `MergeClasses`), contrairement à une somme de changements élémentaires (p.ex. plusieurs `DeleteClass` et un seul `AddClass`), permet aux acteurs de comprendre plus aisément la modification d'une ontologie.

Avantages des changements complexes du point de vue de la signification

- Les changements complexes possèdent une sémantique plus importante que les élémentaires. Par exemple, la sémantique d'un changement complexe de division – `SplitClass` – est clairement plus riche que l'ensemble de changements élémentaires formé d'un seul `DeleteClass` et de plusieurs `AddClass`. De ce fait, les changements complexes peuvent mieux capter l'intention exacte des acteurs ayant fait évoluer l'ontologie et peuvent, par la suite, offrir une vue pertinente sur la vraie signification de l'évolution.

- Les changements complexes permettent d'analyser plus précisément les conséquences sur la conceptualisation de l'ontologie ainsi que les effets sur le référencement sémantique de ressources. Par exemple, en sachant qu'une certaine `classeC` a été fusionnée à l'aide d'un changement `MergeClasses` et qu'un certain nombre de sous-classes de la `classeC` ont été déplacées à la `classeR`, résultante de la fusion, on peut déduire que les ressources référencées par la `classeC` peuvent maintenant être référencées par la `classeR`, puisque celle-ci contient la signification (ou l'intention) de la `classeC`.

Par conséquent, il ne suffit pas d'utiliser uniquement des changements élémentaires, mais il faudrait enrichir le processus d'évolution par l'utilisation des changements complexes. De plus, si le but est de préserver l'intégrité du référencement sémantique de ressources après l'évolution des ontologies, l'utilisation des changements complexes est d'autant plus importante. Cette utilisation permet de rendre explicite la vraie intention sous-jacente à l'évolution et d'analyser les vrais effets des changements sur le référencement des ressources.

3.1.2.3. **Taxonomie de changements complexes et élémentaires**

Les deux auteurs, Stojanovic, (2004), et Klein, (2004), proposent une taxonomie des changements élémentaires respectivement pour le modèle d'ontologies KAON et pour celui d'ontologies OWL. Leur intention était d'offrir une vue exhaustive sur les modifications simples, applicables lors de l'évolution. Les auteurs motivent leur choix par le fait que les changements élémentaires sont les seuls qu'on puisse définir de manière exhaustive (p.ex. en appliquant des méta-changements à chaque élément caractéristique de l'ontologie), contrairement aux changements complexes qu'on ne peut pas décrire exhaustivement puisque le nombre de combinaisons possibles est infini. Il est bien vrai qu'on ne peut décrire l'ensemble des changements complexes, mais il est toutefois possible d'extraire des changements plus typiques et de construire ainsi une hiérarchie de changements complexes à ajouter à la taxonomie des changements élémentaires.

3.1.2.4. **Effets des changements élémentaires et complexes**

On pourrait aussi se questionner sur le fait de savoir si la spécification des changements ne devrait pas contenir une description des conséquences de ces changements sur l'ontologie elle-même ou sur le référencement sémantique de ressources. Bien que les deux auteurs abordent brièvement cet aspect, ils n'incorporent cependant pas ce genre d'information dans les taxonomies présentées. Klein (2004), parle plutôt de changements riches, par exemple l'*élargissement du domaine d'une propriété*, que des effets des changements. De même, Stojanovic (2004), présente, par exemple, la *généralisation d'un concept* ou encore la *spécialisation d'un concept* comme étant de changements complexes et non pas des conséquences.

3.1.2.5. **Spécification de changements influencée par le modèle de l'ontologie**

Finalement, la spécification des changements dépend du modèle de représentation de l'ontologie évolutive : l'ensemble de changements applicables à une ontologie OWL ne peut pas être le même que celui applicable à un autre type d'ontologie (p.ex. DAML+OIL, SCHOE). Même si nous pouvons retrouver des recoupements entre divers modèles de représentation, principalement au sujet des classes et sous-classes, il existe également des différences. Certains modèles parlent de propriétés et axiomes de classes, selon la logique descriptive, d'autres d'attributs et valeurs, selon la logique des *frames*. Certains modèles introduisent même une notion de rôles pour conceptualiser les classes en fonction du domaine d'utilisation (Kozaki, Sunagawa, Kitamura, et Mizoguchi, 2007).

3.2 **Ontologie des changements applicables à des ontologies OWL-DL**

3.2.1. **Construction de l'ontologie de changements**

En partant de la discussion ci-dessus et en tenant compte des taxonomies existantes, nous proposons dans cette section une ontologie des changements applicables à des ontologies OWL-DL. Cette ontologie élargie la conceptualisation de Klein, (2004), par l'ajout d'un nombre de caractéristiques comme : (a) une typologie des changements complexes ; (b) une description des effets de changements sur le référencement sémantique ;

(c) une clarification de la terminologie utilisée pour caractériser la complexité d'un changement.

Précisons cependant que l'ontologie des changements présente une **conceptualisation préliminaire** d'un certain nombre de changements élémentaires et complexes. Étant donné qu'on se situe dans une perspective exploratoire où l'accent est mis sur la conception des solutions partielles, et non sur la conception des produits finalisés, notre intention n'était pas de développer une ontologie exhaustive, mais plutôt perfectible, une ontologie que nous pouvons faire évoluer en tout temps. D'ailleurs, le premier résultat concernait uniquement une classification de changements (Rogozan, 2004a) que nous avons ensuite adaptée et enrichie pour arriver à l'ontologie présentée ci-après.

Précisons également qu'il s'agit d'une **conceptualisation réalisée en parallèle** avec le développement des modules *ChangeHistoryBuilder* et *SemanticAnnotationModifier*. Le fait d'avoir commencé par une petite réalisation, que nous avons ensuite affinée au fur et à mesure que la thèse prenait forme, nous a permis d'orienter la construction de l'ontologie en fonction des études théoriques, mais aussi en fonction de notre propre pratique et expérience accumulées lors de l'avancement dans la conception des systèmes.

3.2.2. **Présentation de l'ontologie des changements**

Dans cette section, nous présentons notre ontologie de changements et nous l'illustrons graphiquement à l'aide de l'éditeur MOT+Onto[25], conçu par une équipe du centre de recherche LICEF[26] de la Télé-université. Pour la représentation graphique des ontologies OWL, cet éditeur utilise des symboles graphiques spécialisés en fonction de la norme OWL-DL. Dans l'Appendice C, le lecteur peut trouver une description détaillée de la représentation graphique à l'aide de l'éditeur MOT+Onto.

[25] MOT+Onto est un éditeur graphique des ontologies OWL, développé au centre de recherche LICEF. Il est accessible à l'adresse `http://www.licef.teluq.uquebec.ca/fr/index.htm`.
[26] Laboratoire d'Informatique Cognitive et d'Environnements de Formation.

Précisons que, dans le but d'alléger la lecture, nous décrivons l'ontologie sans spécifier tous ses détails ni tous les concepts explicités. L'ontologie au complet peut être consultée (en format OWL) à l'adresse : http://www.licef.ca/Ontologie-Changements.

Précisons aussi que nous avons conçu l'ontologie des changements en suivant le modèle de représentation OWL. Le lecteur peut trouver une description détaillée du modèle OWL dans l'Appendice B.

3.2.2.1. **Classes racines de l'ontologie**

L'ontologie se compose des trois classes racines : (1) la classe `ChangeOperation`, qui spécifie les types de changements applicables aux ontologies OWL-DL ; (2) la classe `ChangeObject`, qui présente les objets sur lesquels agissent les changements ; (3) la classe `ChangeEffect`, qui décrit les types de conséquences qu'on peut rencontrer après avoir effectué un changement. Dans cette ontologie, c'est la hiérarchie sous-jacente à la racine `ChangeOperation` qui est la plus riche du point de vue du nombre d'éléments représentés. Les deux autres classes racines, `ChangeObject` et `ChangeEffect`, possèdent un nombre plus réduit d'éléments que nous avons introduits pour permettre une caractérisation plus poussée de changements.

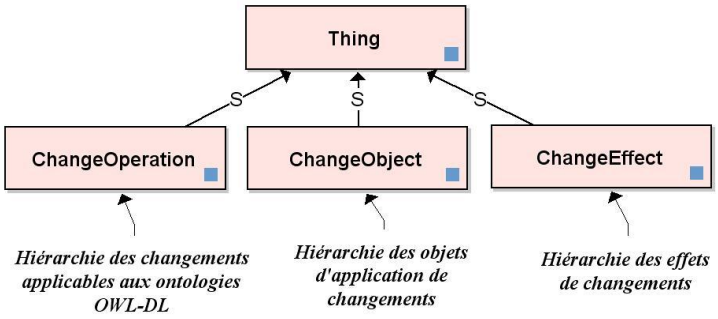


Figure III-2. Classes racines de l'ontologie des changements.
(La légende du formalisme graphique MOT+Onto est consultable dans l'Appendice C)

3.2.2.2. **Hiérarchie de la classe-racine `ChangeObject`**

La classe `ChangeObject` est la racine de la hiérarchie des éléments[27] utilisés pour construire des ontologies OWL comme les éléments de classes et de propriétés, par exemple. Elle est constituée de trois sous-classes (*cf.* Figure III-3) : `ChangeObject_Ontology`, `ChangeObject_Class` et `ChangeObject_Property` (avec ses deux sous-classes `ChangeObject_ObjectProperty` et `ChangeObject_DatatypeProperty`).

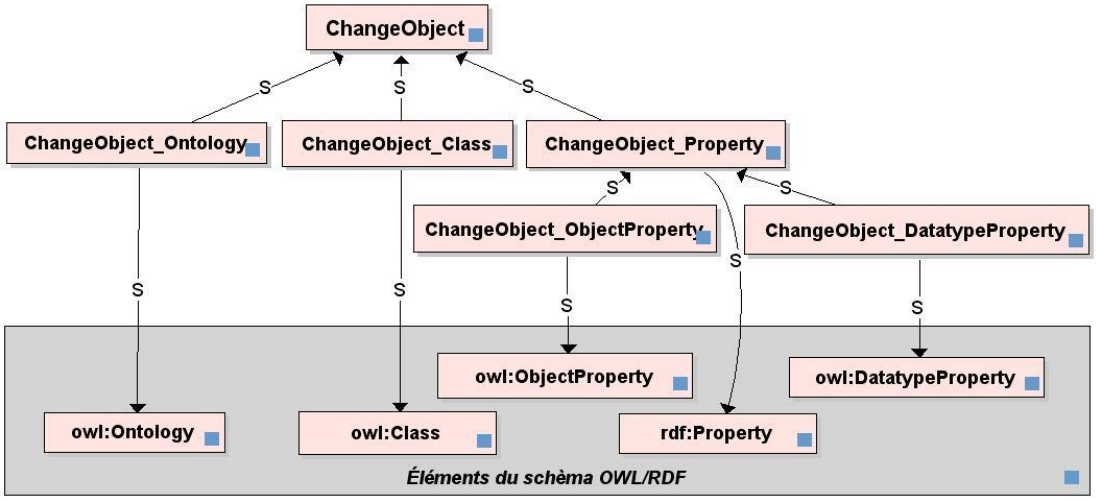


Figure III-3. Hiérarchie `ChangeObject`
(La légende du formalisme graphique MOT+Onto est consultable dans l'Appendice C)

Pour pouvoir utiliser le jeu d'assertions défini déjà dans le schéma standard du langage OWL, nous avons déclaré les sous-classes de `ChangeObject` comme étant des sous-classes des éléments préfixés `owl:Ontology`, `owl:Class` et `rdf:Property` (avec `owl:ObjectProperty` et `owl:DatatypeProperty`). De cette manière, nous avons introduit une relation pour dire, par exemple, que le concept de `ChangeObject_Class` est une sorte de `owl:Class` et qu'il hérite ainsi de toutes les caractéristiques de cet élément OWL.

Les instances de classes et de propriétés sont, elles aussi, des éléments utilisés dans la construction des ontologies OWL. Cependant, pour orienter, dans un premier temps, la conceptualisation de l'ontologie sur les changements qui impliquent les classes et les

[27] Le lecteur pourra trouver une description détaillée des éléments OWL dans l'Appendice B.

propriétés, nous ne les avons pas traitées dans ce travail. Une fois cette conceptualisation stabilisée, on pourrait enrichir l'ontologie par l'ajout de changements applicables à des instances de classes et à des instances de propriétés.

3.2.2.3. **Hiérarchie de la classe-racine `ChangeEffect`**

Les changements effectués lors du processus d'évolution peuvent avoir différents effets sur : (1) le référencement sémantique de ressources; (2) la relation logique entre les classes de V_N et celles de V_{N+1}. Ces effets seront discutés en détail dans le chapitre V. Dans cette partie, nous résumons brièvement la hiérarchie `ChangeEffect` et nous la mettons en relation avec les types de changement, le but étant d'offrir une vue complète sur l'ontologie de changements, même si certains concepts abordés ne trouvent qu'un peu plus loin leur entière signification.

Tel qu'illustré dans la Figure III-4, la classe racine `ChangeEffect` possède deux sous-classes. La sous-classe `ChangeEffect_On_LogicalRelation` spécifie la relation logique entre une classe de V_N et la même classe ou une autre appartenant à la V_{N+1}, selon des critères comme l'équivalence, l'inclusion, la généralisation[28], la spécialisation ou encore la différence conceptuelle. Elle représente alors un moyen de caractériser la dimension conceptuelle de V_{N+1} et de souligner ainsi l'impact des changements sur la signification des classes ayant été modifiées lors d'un processus d'évolution.

[28] Par exemple, une classe peut devenir plus générale en V_{N+1} si on lui a effacé un axiome. Elle devient, par contre, plus spécialisée si on lui ajoute un axiome.

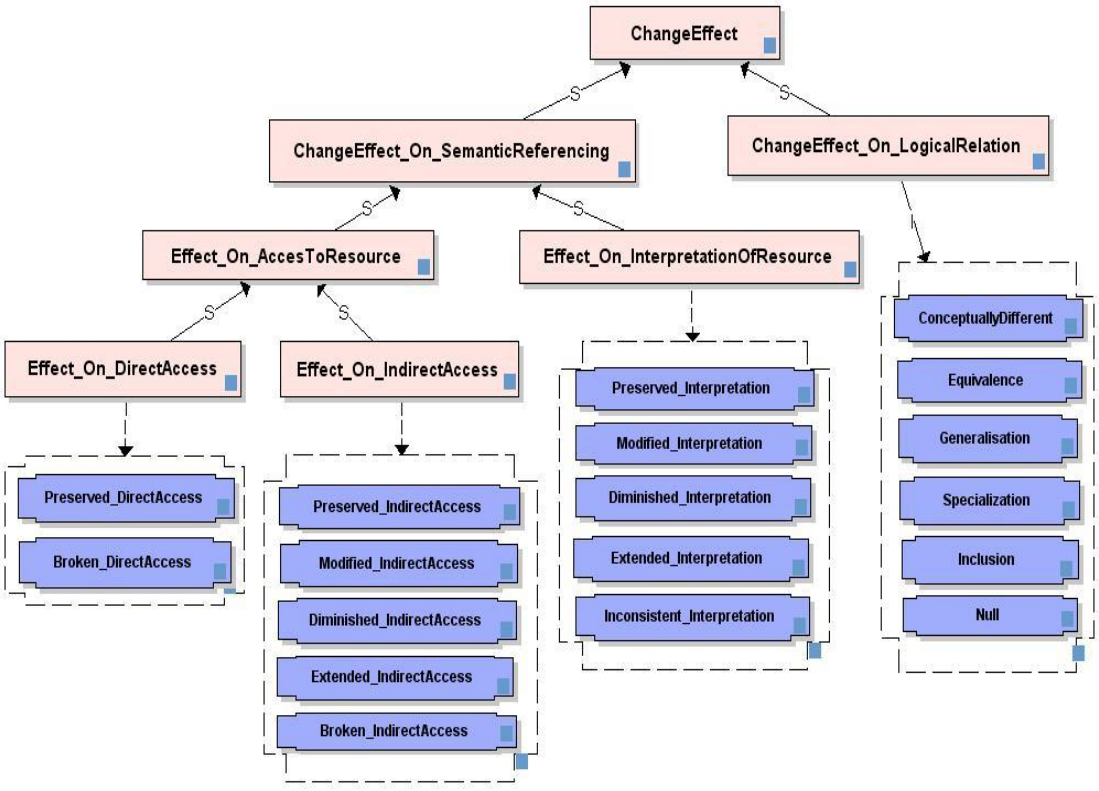


Figure III-4. Hiérarchie `ChangeEffect` et ses instances
(La légende du formalisme graphique MOT+Onto est consultable dans l'Appendice C)

La sous-classe `ChangeEffect_On_SemanticReferencing` réfère aux effets de changements sur la relation entre les ressources et leurs références sémantiques qui sont des éléments d'une ontologie. Cette relation peut en être une d'accès aux ressources, d'où la notion de `ChangeEffect_On_AcessToResource`. À son tour, cet accès peut être direct ou indirect, en fonction de la requête utilisée pour y accéder. La relation peut en être aussi une d'interprétation, sachant qu'une ressource est référencée dans le but d'avoir accès à une description formelle de son contenu et de sa signification. Par conséquent, les effets de changements peuvent se ressentir également au niveau interprétatif, d'où la notion de `ChangeEffect_On_InterpretationOfResource`.

Les classes de type `ChangeEffect` possèdent chacune un ensemble d'instances, correspondant à des effets concrets soit sur l'accès aux ressources, soit sur l'interprétation de celles-ci.

3.2.2.4. **Hiérarchie de la classe-racine `ChangeOperation`**

La hiérarchie de changements (*cf.* Figure III-5), dont la classe racine est `ChangeOperation`, se compose d'un ensemble de changements élémentaires, spécifié par la sous-hiérarchie `ElementaryChange`, ainsi que d'un ensemble de changements complexes, spécifié par la sous-hiérarchie `ComplexChange`. Les deux hiérarchies ont été déclarées disjointes et, par héritage, chacun des changements élémentaires et complexes.

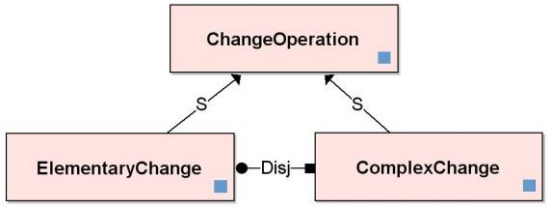


Figure III-5. Les deux sous-hiérarchies de `ChangeOperation`
(La légende du formalisme graphique MOT+Onto est consultable dans l'Appendice C)

3.2.2.4.1. *Opérations de changements élémentaires*

La hiérarchie des changements élémentaires est présentée dans la Figure III-6. Elle contient les changements génériques `Add_Change` et `Delete_Change`, chacun d'entre eux étant ensuite décomposé selon qu'on l'applique à une ontologie (`Add_To_Ontology`, `Delete_From_Ontology`), à une classe (`Add_To_Class`, `Delete_From_Class`) ou à une propriété (`Add_To_Property`, `Delete_From_Property`). Rappelons que les trois objets d'application, c'est-à-dire ontologie, classe et respectivement propriété, sont ceux exprimés dans la hiérarchie `ChangeObject` et possèdent alors les caractéristiques spécifiques du modèle OWL.

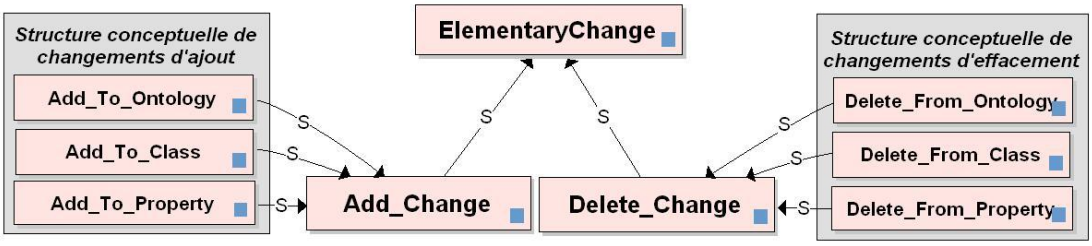


Figure III-6. Hiérarchie des changements élémentaires
(La légende du formalisme graphique MOT+Onto est consultable dans l'Appendice C)

Nous observons que les deux changements génériques, `Add_Change` et `Delete_Change`, possèdent une structure conceptuelle semblable. Nous illustrons dans la Figure III-7 uniquement la classification des changements d'ajout en précisant que les changements d'effacement sont classifiés d'une manière semblable.

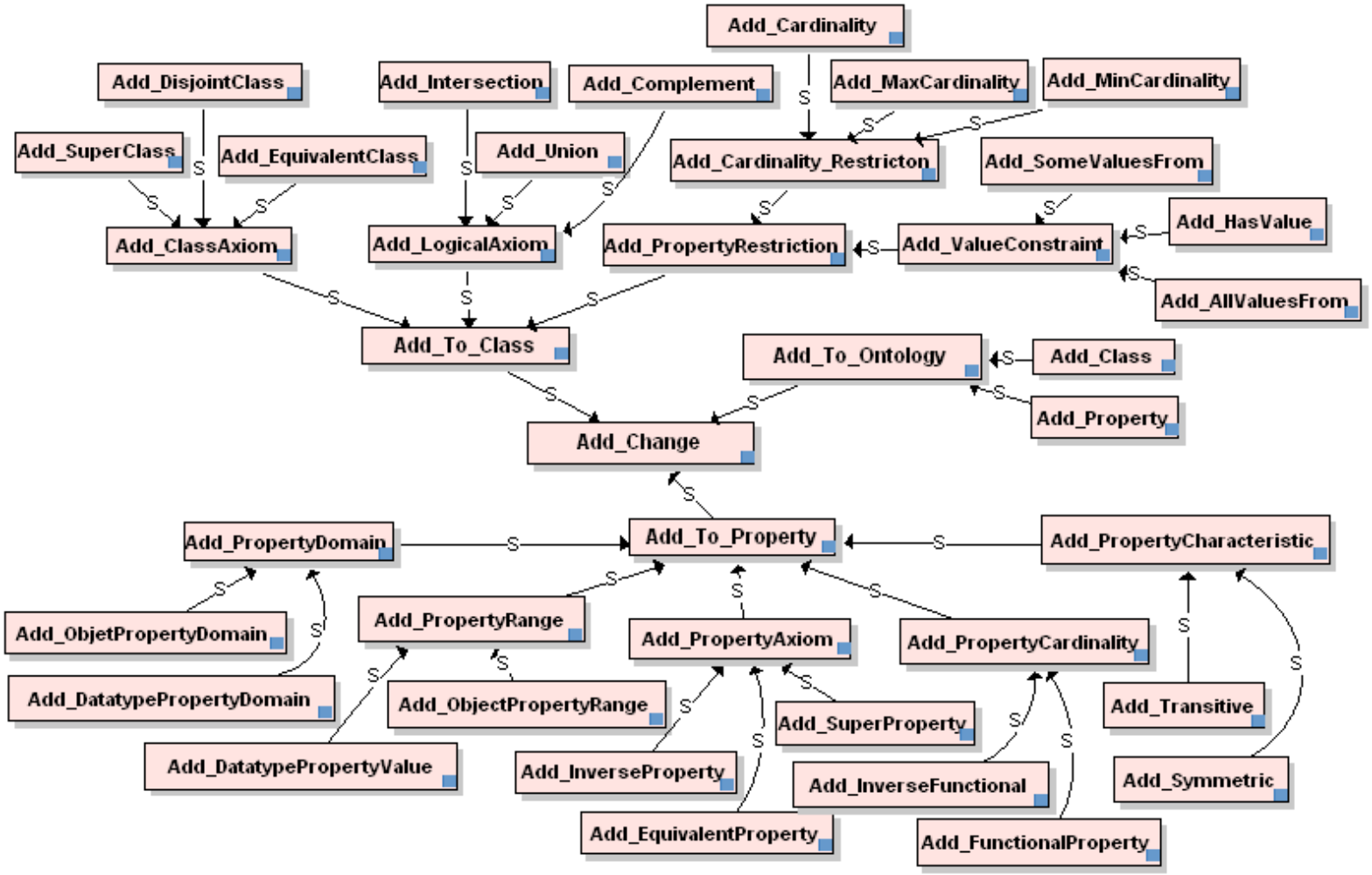


Figure III-7. Classification des changements d'ajout (les changements d'effacement sont classifiés d'une manière semblable)
(La légende du formalisme graphique MOT+Onto est consultable dans l'Appendice C)

Ainsi, du point de vue de l'ontologie, on a les changements `Add_Class` et `Add_Property`, `Delete_Class` et `Delete_Property`. Du point de vue des classes, par contre, les changements se multiplient : on peut avoir l'ajout ou l'effacement d'un axiome logique (`intersectionOf`, `complementOf`, `unionOf`), d'un axiome de classe (`superClass`, `equivalentClass`, `disjointWith`) ou même d'une restriction de classe (`restriction`). Quant aux propriétés, les principaux changements opèrent sur le domaine et le codomaine, ainsi que sur les axiomes de propriétés (`superProperty`, `equivalentProperty`, `inverseProperty`).

3.2.2.4.2. *Opérations de changements complexes*

Un changement complexe est une collection ordonnée de changements élémentaires qui forment ensemble une seule et unique entité logique. Ainsi, chaque changement complexe est issu de la combinaison de deux ou plusieurs changements élémentaires. Ceci fait que l'ensemble de changements complexes est théoriquement infini, puisque le nombre de combinaisons possibles est, lui aussi, théoriquement infini.

Nous pouvons cependant extraire les combinaisons plus typiques, ou les plus utilisables, et les organiser taxonomiquement afin d'obtenir une hiérarchie de changements complexes non pas exhaustive, mais **évolutive** au sens que des changements nouveaux peuvent être nécessaires (issus, par exemple, d'une étude impliquant des sujets humains qui modifient des ontologies) ou que d'autres deviennent désuets.

Nous présentons une partie de la hiérarchie de changements complexes dans la Figure III-8.

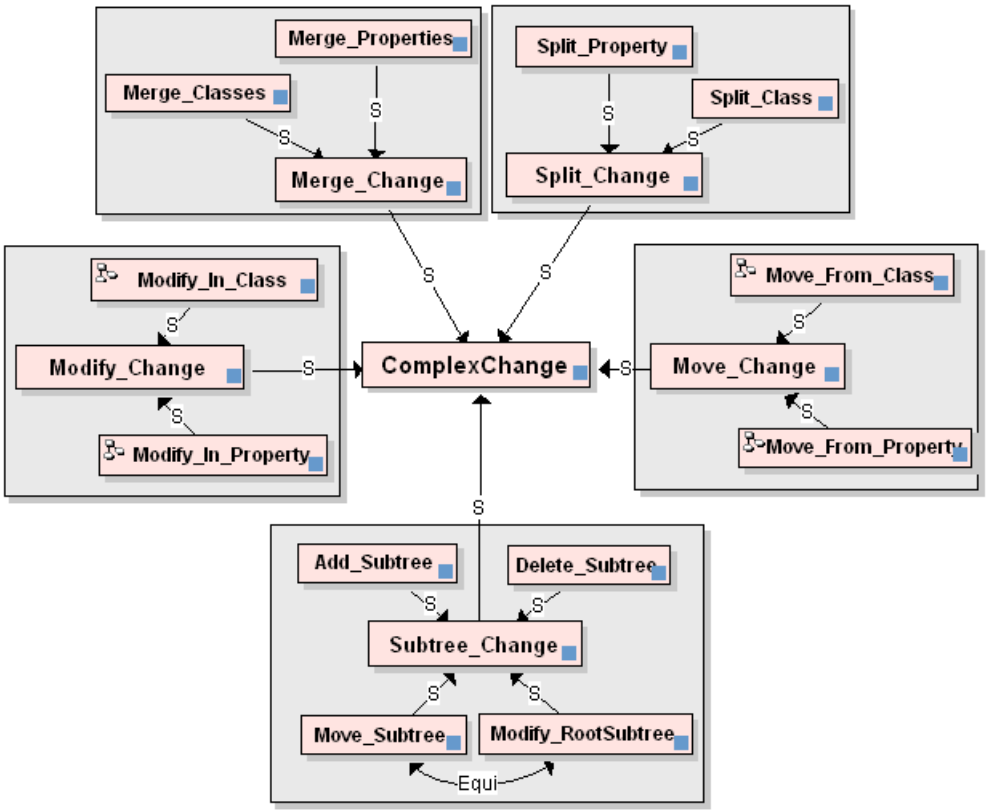


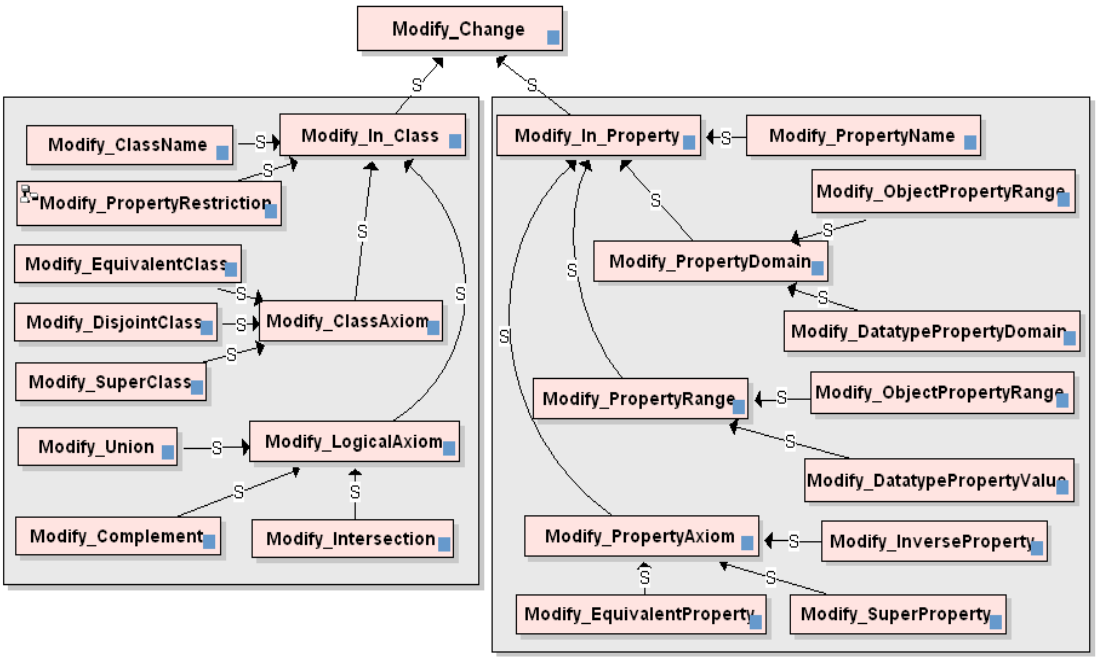
Figure III-8. Hiérarchie des changements complexes

(Le signe attaché à une classe signifie qu'elle possède un sous-modèle descendant, qui n'est pas affiché dans le modèle principal de la représentation ;
La légende du formalisme graphique MOT+Onto est consultable dans l'Appendice C)

Les principaux types de changements complexes sont ceux qui fusionnent, qui divisent, qui modifient ou qui déplacent des éléments (`Merge_Change`, `Split_Change`, `Modify_Change`, `Move_Change`). Nous avons aussi ajouté le `Subtree_Change` afin d'exprimer des changements qui ajoutent ou qui effacent une sous-hiérarchie des classes, ou encore qui modifient sa classe racine.

Les changements de type fusion et division ont chacun deux sous-classes : `Merge_Classes` et `Merge_Properties`, respectivement `Split_Class` et `Split_Property`. Les deux autres types de changements ont, cependant, un nombre plus élevé de sous-classes pour exprimer, par exemple, la modification du nom d'une classe ou d'une propriété, ou la modification des axiomes à l'intérieur d'une classe ou d'une propriété, ou encore le déplacement des axiomes d'une classe à une autre. Dans la Figure III-9, nous

illustrons la classification des changements de type `Modify_Change`, en précisant que les changements de type `Move_Change` sont classifiés d'une manière semblable.



III-9. Classification des changements de type `Modify_Change`
(La légende du formalisme graphique MOT+Onto est consultable dans l'Appendice C)

3.2.2.5. **Propriétés associées aux classes de type `ChangeOperation`**

Les propriétés de l'ontologie de changements concernent des propriétés d'objet et de type de donnée. Dans cette section nous présentons ces propriétés en précisant leur signification générale ainsi que leur domaine et codomaine (*cf.* Figure III-10).

Les **propriétés d'objet** sont au nombre de quatre. La propriété `appliedOn` relie les opérations de changement aux objets d'ontologie auxquels ils s'appliquent, à savoir si un changement apporte des modifications à la structure de l'ontologie, où plutôt à la définition d'une classe ou d'une propriété. Pour décrire chaque opération de changement, les propriétés `haveSource` et `haveTarget` permettent de spécifier les éléments qui composent l'origine d'un changement, et respectivement la cible du changement. Les deux propriétés ont comme domaine une opération de changement (`ChangeOperation`) et comme codomaine un ou

plusieurs éléments de type `ChangeObject`[29] ou des valeurs `rdfs:Datatype` (pour les changements qui impliquent des propriétés de type de données). Finalement, nous avons aussi spécifié la propriété `havePossibleEffect` qui relie une opération de changements aux *effets possibles*[30], spécifiés dans la hiérarchie `ChangeEffect`.

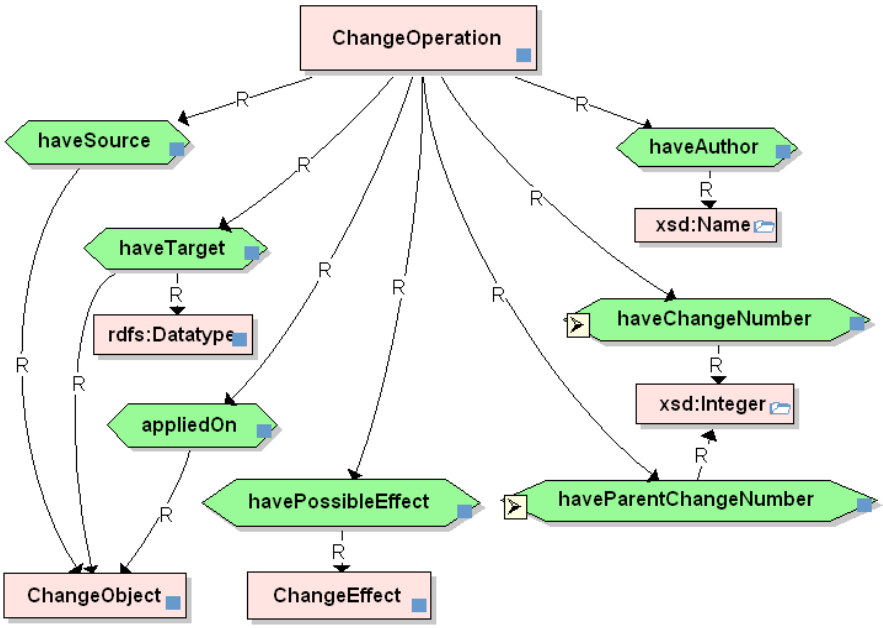


Figure III-10. Propriétés associées aux opérations de changement
(La légende du formalisme graphique MOT+Onto est consultable dans l'Appendice C)

Les **propriétés de type de données** sont au nombre de trois : `haveChangeNumber`, qui spécifie l'identifiant numérique correspondant à l'ordre d'application des changements lors du processus d'évolution ; `haveParentChangeNumber`, qui spécifie l'identifiant numérique du changement-parent, s'il existe ; `haveAuthor`, qui permet de déclarer l'auteur d'un changement. Les deux premières propriétés sont fonctionnelles, puisqu'elles ne peuvent

[29] Les éléments de type `ChangeObject` qui composent le codomaine de `haveSource` et `haveTarget` sont particulièrement des éléments de type `ChangeObject_Class` ou `ChangeObject_Property`. Ces aspects seront cependant spécifiés par des restrictions associées aux changements.
[30] Nous parlons ici bien de l'effet possible, au sens qu'un changement peut avoir un certain effet, mais ceci n'est pas obligatoire. Par exemple, si on ajoute un axiome de disjonction entre une classe A et une autre classe B cela peut rendre inconsistante l'interprétation d'une ressource R référencée, en même temps, par A et B. Mais si la ressource R est référencée uniquement par A, l'ajout de l'axiome de disjonction ne rend pas son interprétation inconsistante. De même, si on efface une classe C qui n'est utilisée dans le référencement d'aucune ressource, cet effacement n'aura aucun effet problématique.

avoir qu'un seul (et unique) identifiant pour chacun des changements. Par contre, les auteurs peuvent être multiples, c'est pourquoi nous n'avons mis aucune restriction sur la troisième propriété de type de données.

3.2.2.6. **Description des changements au moyen des propriétés**

Observez que, dans la section 3.2.2.4. , nous avons seulement précisé le nom des changements (c.-à-d. les étiquettes de classes) ainsi que leur organisation taxonomique. Il est possible cependant d'étendre la définition des changements pour inclure des notions supplémentaires au moyen des restrictions de propriétés. Ainsi, la plupart des définitions des changements se composent maintenant de deux parties : une introduction (ou un appel) de nom et une liste de restrictions spécifiques ou héritées de classes supérieures.

3.2.2.6.1. *Description de changements élémentaires*

Dans la Figure III–11, nous présentons un extrait de la description des changements élémentaires de type `Add_Change`, une description réalisée à l'aide de restrictions sur des propriétés. Pour tous les changements d'ajout nous avons introduit deux restrictions générales : une cardinalité exacte de 0 pour la propriété `haveSource`, qui dit que la source d'un changement d'ajout ne se compose d'aucun élément ; une cardinalité minimale de 1 pour la propriété `haveTarget` qui dit que la cible d'un changement d'ajout se compose au moins d'un élément.

De plus, pour chaque type d'ajout, nous avons spécifié si la cible du changement concerne des classes, des propriétés ou des données RDF, en ajoutant des contraintes `owl:allValuesFrom` sur la propriété `haveTarget`.

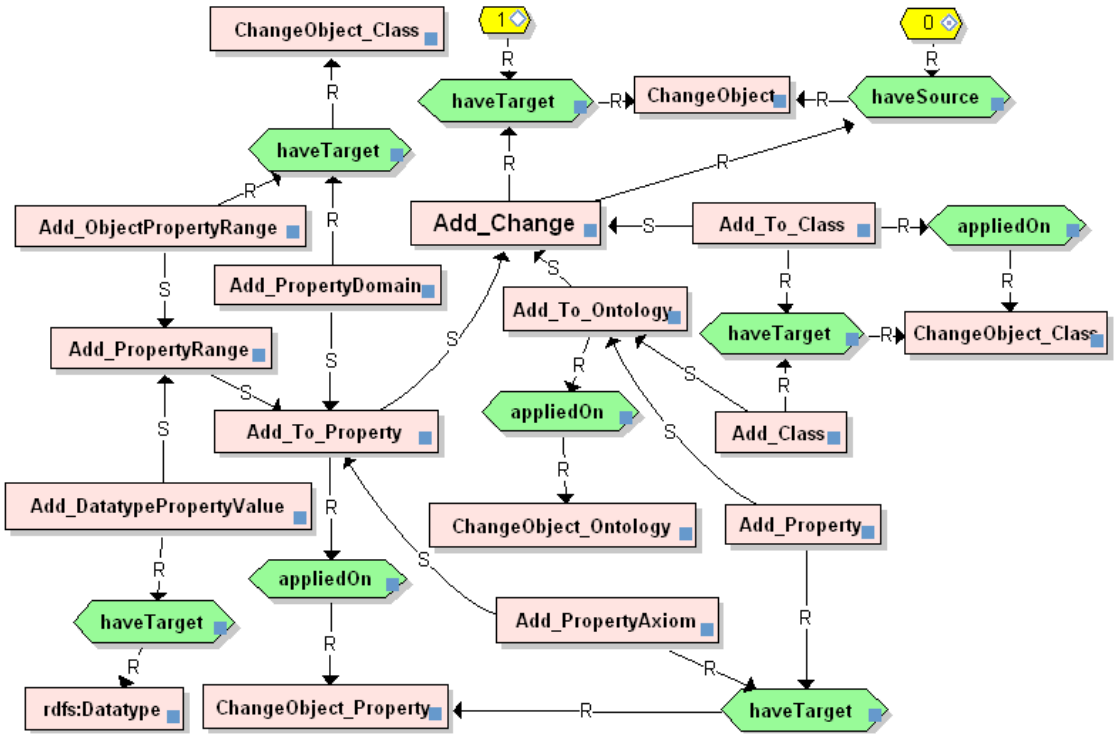


Figure III-11. Extrait de la description des changements de type `Add_Change`
(La légende du formalisme graphique MOT+Onto est consultable dans l'Appendice C)

Nous avons également précisé l'objet d'application pour chaque changement en leur ajoutant trois restrictions de la propriété `appliedOn`. Ceci pour spécifier les changements qui ajoutent des éléments à la structure de l'ontologie (`Add_To_Ontology`), à la définition d'une classe (`Add_To_Class`) ou à la définition d'une propriété (`Add_To_Property`). Le fait de préciser l'objet d'application d'un certain changement nous permet de mieux spécifier ses effets. Par exemple, un changement qui s'applique à une ontologie, comme l'effacement ou la fusion de classes, modifie la structure conceptuelle de l'ontologie et peut ainsi entraîner des effets sur l'accès aux ressources référencées. Quant à une opération qui s'applique à une classe, comme l'ajout d'un axiome de disjonction ou la modification d'une superclasse, elle change la définition de la classe en question et peut avoir un impact sur l'interprétation des ressources référencées par celle-ci.

Pour les changements de type `Delete_Change` nous avons utilisé le même raisonnement que celui présenté pour `Add_Change`. Les seules différences concernent les

restrictions de cardinalités - pour `haveSource` (minimum 1) et `have Target` (exactement 0), comme illustré dans la Figure III-12.

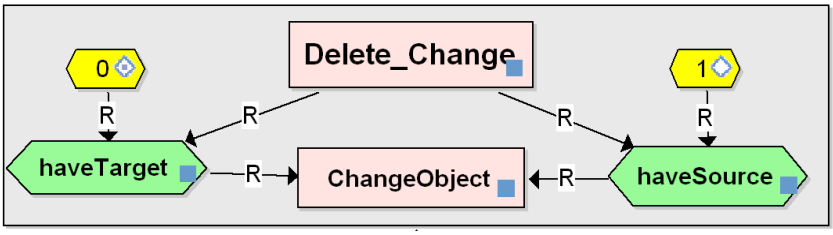


Figure III-12. Extrait de la description de changements de type `Delete_Change`
(La légende du formalisme graphique MOT+Onto est consultable dans l'Appendice C)

3.2.2.6.2. *Description de changements complexes*

Pour les changements complexes, nous avons mis en œuvre la même démarche que pour les changements élémentaires. Nous l'illustrons dans la Figure III-13, en considérant les changements `Merge_Change` et le `Split_Change` (pour la plupart des autres changements complexes, nous avons suivi un raisonnement semblable).

Ainsi, nous avons commencé par déclarer le nombre d'éléments qui constituent la source et la cible d'un changement complexe en ajoutant des restrictions de cardinalité sur les propriétés `haveSource` et `haveTarget`. Par exemple, les changements de type `Merge_Change` doivent avoir au minimum deux éléments dans la source du changement (exactement un pour le `Split_Change`) et exactement un élément dans la cible du changement (minimum deux pour le `Split_Change`). Ensuite, nous avons spécifié si la source et la cible du changement se composent des classes, des propriétés ou des deux. Nous avons aussi précisé si les changements apportent des modifications à l'ontologie, à une classe ou à une propriété en particulier, en ajoutant des restrictions sur la propriété `appliedOn`.

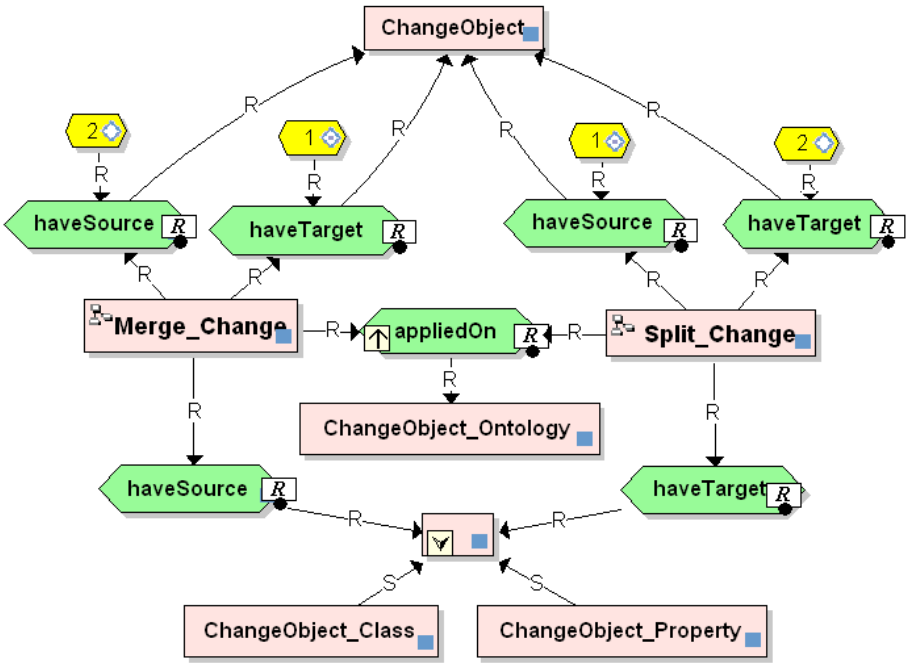


Figure III-13. Description des changements complexes `Merge_Change` et `Split_Change`
(La légende du formalisme graphique MOT+Onto est consultable dans l'Appendice C)

3.2.2.6.3. *Description des effets de changements élémentaires et complexes*

Pour déclarer les effets des changements, nous avons utilisé des restrictions de type 'a comme valeur[31]' sur les valeurs de la propriété `havePossibleEffect`. À chaque changement, nous avons associé des valeurs précises, provenant, par exemple, de l'ensemble des instances de la classe `Effect_On_DirectAccess` (pour dire qu'un changement puisse ou non invalider l'accès direct[32] à une ressource référencée) ou de l'ensemble des instances de la classe `Effect_On_IndirectAccess` (pour dire qu'un changement puisse invalider, élargir, diminuer ou préserver l'accès indirect à une ressource référencée). Dans la Figure III–14, nous présentons quelques exemples de cette démarche.

[31] Il s'agit de la restriction `hasValue` qui décrit la classe pour laquelle la propriété spécifiée a au moins une valeur sémantiquement égale à la valeur désignée.
[32] Les définitions reliées aux effets de changements sont données dans le chapitre V.

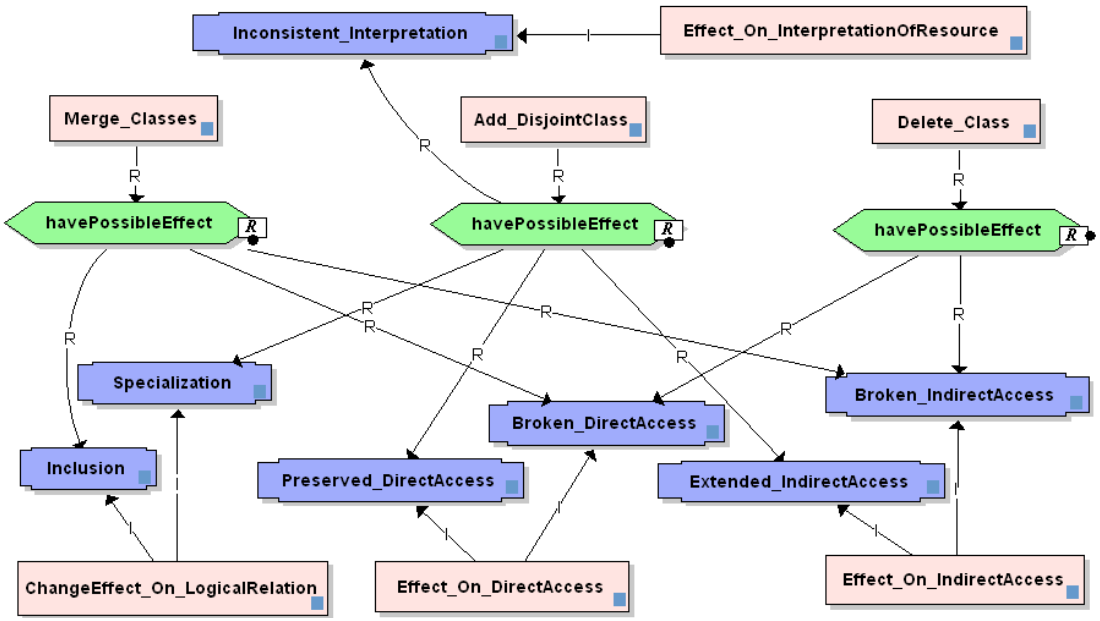


Figure III-14. Effets de changements élémentaires et complexes : trois exemples
(La légende du formalisme graphique MOT+Onto est consultable dans l'Appendice C)

3.2.3. **Formalisation de l'ontologie en OWL**

Rappelons que l'ontologie des changements a comme but de décrire les types de changements possibles, mais aussi de donner aux agents informatiques (particulièrement aux CHB et SAM) la possibilité d'exploiter cette description. Pour la formalisation de l'ontologie, nous avons eu recours au langage standard OWL33. Dans la Figure III–15[34], nous montrons un extrait de l'ontologie des changements, formalisée en OWL.

[33] L'exportation de l'ontologie en format OWL est réalisée automatiquement par l'éditeur MOT+Onto.
[34] Rappelons que l'ontologie de changements (en format OWL) est accessible à l'adresse suivante : http://www.licef.ca/Ontologie-Changements.

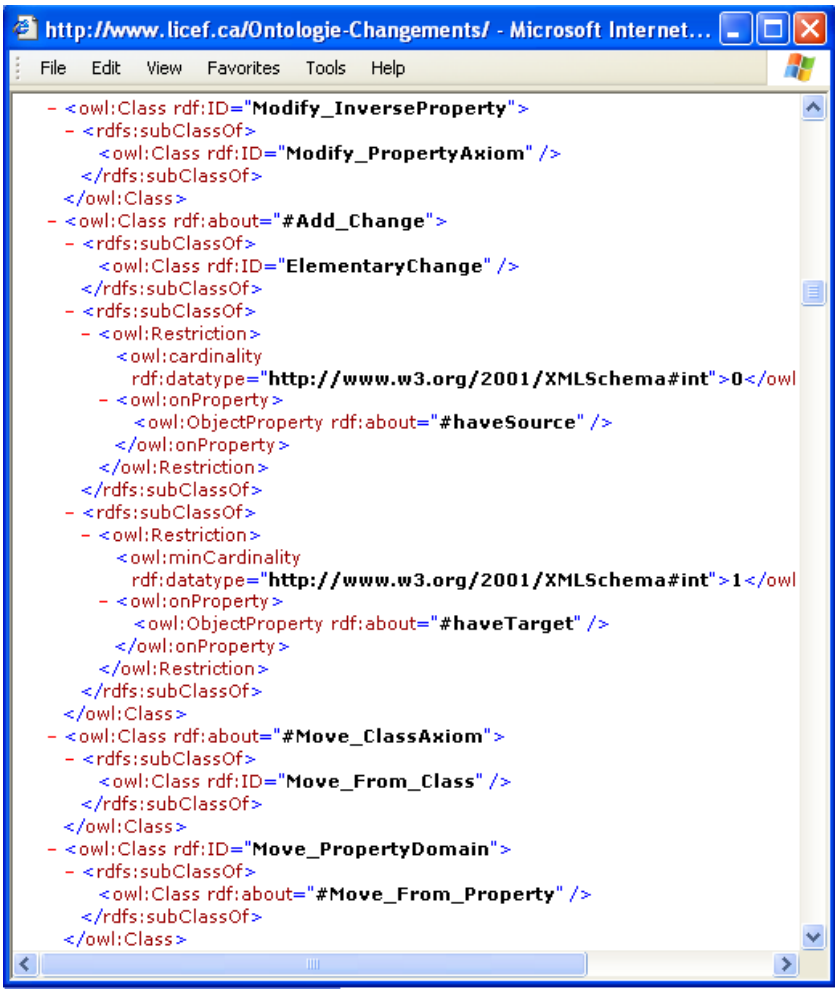


Figure III-15. Extrait de l'ontologie des changements, formalisée en OWL

3.3 **Conclusion**

Dans ce chapitre nous avons présenté l'ontologie des changements que nous avons développée pour fournir une spécification formelle des changements complexes ou encore une spécification plus complète des changements élémentaires.

Notre ontologie se compose de deux hiérarchies de base, celle des changements élémentaires (environ 50 concepts) et celle des changements complexes (environ 40 concepts). Elle propose deux autres hiérarchies (`ChangeObject` et `ChangeEffect`), que

nous avons utilisées pour une caractérisation plus poussée des changements. Nous avons réalisé cette caractérisation en utilisant également un certain nombre de restrictions de propriétés décrivant la source et la cible du changement (`haveSource` et `haveTarget`), son objet d'application (`appliedOn`) mais aussi l'effet de chaque changement en particulier (`havePossibleEffect`). Finalement, nous avons formalisé l'ontologie des changements en OWL pour la rendre exploitable par tout agent informatique compatible avec ce langage.

Rappelons que cette ontologie spécifie une conceptualisation préliminaire, perfectible et essentiellement évolutive, que nous avons définie au fur et à mesure que nous avancions dans le développement des systèmes exploratoires CHB et SAM.

L'ontologie des changements est un aspect important du cadre de référence que nous proposons pour la gestion des changements apportés aux versions d'ontologie. La conceptualisation qu'elle spécifie est utilisée autant pour la journalisation des changements avec le système CHB que pour la modification du référencement affecté par les changements, modification supportée par le système SAM. Nous présenterons ces deux systèmes dans le chapitre IV et dans le chapitre V.

Chapitre IV

IDENTIFICATION DES CHANGEMENTS APPORTÉS AUX VERSIONS D'ONTOLOGIE

La gestion de l'historique des changements est un aspect primordial de l'évolution des ontologies. Sans connaître les changements apportés aux versions d'ontologie, on ne peut connaître ni la signification de l'évolution, ni ses effets sur l'intégrité du référencement sémantique des ressources. Dans ce chapitre, nous présentons le premier module informatique qui compose notre cadre de référence pour la gestion des changements : le système *ChangeHistoryBuilder* (CHB) pour la journalisation et l'identification des changements apportés aux versions d'ontologie réparties sur le Web.

Dans la Section 4.1, nous faisons l'état de l'art des approches de gestion de l'historique des changements et nous mettons en évidence leurs limites. Nous proposons ensuite le système CHB (*cf.* Section 4.2) comme une solution pour dépasser les limites reliées aux problèmes d'accès et d'interprétation de cet historique, mais aussi celles concernant l'identification des changements complexes, en plus des changements élémentaires. Nous précisons que le fonctionnement du CHB est fondé, en partie, sur l'ontologie des changements. Nous concluons ce chapitre en résumant les apports du CHB et en démontrant leur palette d'applicabilité (*cf.* Section 4.3).

Les travaux présentés dans ce chapitre ont donné lieu à deux publications (Rogozan et Paquette, 2005a; Rogozan, Paquette, et Rosca, 2004).

4.1 **Identification des changements apportés aux ontologies : état de l'art**

Dans le contexte du Web sémantique, une tâche essentielle est celle de tracer et d'identifier les changements apportés aux versions d'ontologie d'une manière riche, mais également rapide. Les calculs effectués pour identifier les changements doivent prendre peu du temps et les réponses doivent parvenir rapidement aux utilisateurs et agents du web. Dans les sections suivantes, nous faisons un état de l'art des méthodes et des outils, en mettant en évidence leurs apports et leurs limites respectives.

Actuellement, on peut distinguer deux approches de traçage et d'identification des changements apportés aux ontologies. La première est fondée sur la journalisation des changements, c'est-à-dire l'enregistrement, dans un fichier-journal (habituellement un fichier texte), des opérations effectuées par les utilisateurs lors de la modification de l'ontologie. Ceci en vue de garder une trace de changements ontologiques. La deuxième est fondée sur la comparaison de versions d'ontologies au niveau structurel et l'utilisation d'un ensemble des règles en vue d'identifier les modifications.

4.1.1. **Journalisation des changements**

Commençons la discussion des approches existantes en faisant une distinction entre (1) les modèles conceptuels (sans aucune implémentation technologique) qui développent une théorie portant sur la description des changements et (2) les outils capables de supporter effectivement la journalisation des changements lors du processus d'évolution.

4.1.1.1. **Modèles conceptuels de description des changements**

4.1.1.1.1. *Le modèle CONCORDIA*

Oliver, Shahar, Musen et Shortliffe, (1999), proposent le modèle CONCORDIA pour la gestion des changements d'une terminologie médicale. Ce modèle associe à chaque classe

un identifiant unique sous la forme d'une chaîne de caractères alphanumériques. Pour garder trace des classes et attributs ajoutés, renommés ou retirés, ce modèle propose un ensemble des schémas pour spécifier les données à être enregistrées dans des fichiers-journal lors de l'exécution des changements : (1) des données générales comme la date, l'auteur et une explication en langage naturel et (2) des données propres à chaque changement, comme l'identifiant et le nom de la sous-classe ajoutée pour le changement `AddChild` ou encore comme l'identifiant et l'ancien et le nouveau nom pour `ReplaceAttributeName`.

Tableau IV-1 Exemples de schémas-type de description des changements

AddChild	**ReplaceAttributeName**	**RetireConcept**
Date, Auteur, Explication		
Identifiant unique de concept Nom du concept courant Identifiant unique du nouveau concept-enfant Nom du nouveau concept-enfant	Identifiant unique de l'attribut Nom de l'attribut Nouveau nom de l'attribut	Identifiant unique du concept retiré Nom du concept retiré

CONCORDIA offre ainsi un modèle de description des changements qui, en plus de mettre en évidence les données à retenir lors du processus d'évolution, offre un cadre organisé pour l'archivage des changements ontologiques.

Discussion du modèle CONCORDIA

Chaque schéma de description possède une structure interne qui lui est propre, composée de descripteurs propres à chaque changement. Dans le modèle CONCORDIA, c'est bien la sémantique d'un changement qui impose la structure syntaxique du schéma, ce qui conduit à une diversité structurelle dont la gestion peut s'avérer fastidieuse dans le contexte réparti et décentralisé du web.

Le modèle CONCORDIA propose des schémas pour journaliser un bon nombre de changements élémentaires. À notre avis, certains cas sont toutefois traités d'une manière redondante. Par exemple, `ReplaceConceptName` et `CorrectConceptName` sont tous les deux des changements qui modifient le nom d'une classe, avec la seule différence que le premier conserve l'ancien nom sous la forme d'un synonyme contrairement au dernier. Ceci pourrait alors conduire à un processus de journalisation trop spécialisé, où chaque petite différence conduit à un changement enregistré selon un schéma différent.

Les schémas ont été développés sur la base de quelques hypothèses simplificatrices. Par exemple, celles qui supposent que toutes les sous-classes seront automatiquement reliées à la classe-parent de la classe retirée ou encore celle qui suppose que tous les attributs de la classe retirée seront également retirés. Par conséquent, ces schémas ne permettent pas le traitement des situations ouvertes, où ce sont les acteurs du processus d'évolution qui choisissent la stratégie à suivre pour garder la consistance de l'ontologie (*cf.* Section 2.2.2.). Ils ne peuvent journaliser donc que des cas bien spécifiques.

Le modèle CONCORDIA ne propose aucun format pour enregistrer, stocker ou échanger les schémas de description d'une manière uniforme et compatible avec un langage standard de représentation d'ontologies. Dans le contexte du Web, la représentation des changements dans un langage ayant une sémantique formelle est néanmoins une nécessité, si le but est de permettre aux agents logiciels de raisonner dessus.

4.1.1.1.2. *Autres modèles de description des changements*

Un autre modèle de description des changements est celui proposé par Ognyanov et Kiryakov, (2002) . Ces auteurs développent une approche pour documenter seulement deux changements : ceux qui effacent et ceux qui ajoutent des triplets RDF à un ensemble de déclarations RDF qui forment une ontologie. Le peu de changements traités est justifié par le fait qu'un autre type de changement, par exemple la modification d'une déclaration, peut être vu comme une séquence d'ajouts ou d'effacements des triplets RDF. Si syntaxiquement parlant cela est valide, sémantiquement parlant cette séquence est loin du sens exact du changement de modification. Concernant les changements d'une complexité plus grande, comme la fusion ou la séparation, ils ne sont pas du tout abordés dans cette approche.

4.1.1.2. **Outils de support à la journalisation de changements ontologiques**

4.1.1.2.1. *La suite d'outils de KAON*

KAON, une suite d'outils pour les ontologies et le Web sémantique, est actuellement le seul système de gestion d'ontologies qui possède une fonction dédiée à la journalisation des changements (Maedche, Motik, et Stojanovic, 2003; Stojanovic, 2004). Lors de

l'évolution, KAON enregistre dans un fichier-journal, sous la forme d'une séquence ordonnée de déclarations RDF/XML, tous les changements exécutés pour passer d'une version de l'ontologie à une autre. Ces changements sont classifiés en fonction d'un modèle `a:ChangeLog` qui spécifie des changements additifs de type `a:AddEnitity` ou soustractifs de type `a:RemoveEntity`. Pour expliciter formellement l'enchaînement des changements, ces derniers sont reliés, à l'aide d'une propriété `a:hasPreviousChange`, aux changements antérieurs. Quant à la propriété `a:hasPreviousHistoryChange`, elle permet de formaliser la relation de causalité qui existe entre un changement initial, nommé `a:hasRequestedChange`, et ses changements additionnels. Un exemple du fichier-journal du KAON est fourni dans le Tableau IV-2.

Discussion du processus de journalisation supporté par KAON

Le processus de journalisation supporté par KAON est le plus complet à l'heure actuelle. Il permet : (a) l'enregistrement des changements élémentaires selon le modèle `a:ChangeLog`, fondé sur l'ontologie des changements élémentaires de Stojanovic (2004) ; (b) la représentation explicite du lien de dépendance existant entre des changements primaires et additionnels. Toutefois, le processus de journalisation supporté par KAON possède quelques limitations importantes.

La première limitation est que le système KAON ne traite pas les changements complexes, ce qui est une forte limitation si le but est de formaliser le plus précisément possible l'intention, le sens donné par l'utilisateur à un changement ou à une séquence de changements. Considérons, par exemple, une séquence de changements explicitée formellement dans le fichier-journal du KAON (*cf.* Tableau IV-2). Cette séquence se compose d'un nombre de changements qui effacent les classes `Professeur`, `Tuteur` et `Animateur` et d'autres qui ajoutent les classes `FacilitateurApprentissage` et `AnimateurForum`. Malheureusement, même si cette séquence précise une information sur les changements accomplis, elle ne reflète que partiellement la signification réelle, c'est-à-dire celle donnée par l'acteur ayant fait évoluer l'ontologie. Elle peut même refléter une signification assez différente de cette signification réelle.

Tableau IV-2 Journal du Kaon et signification extraite *versus* réelle

Fichier-journal du KAON	Signification extraite du journal KAON	Signification réelle
(1) <a:RemoveConcept rdf:ID="i-1104860243" a:has_referenceConcept="testOntology#Professeur" a:inOIModel="testOntology" a:version="85"> ... </a:RemoveConcept> (2) <a:RemoveConcept rdf:ID="i-1079545047999-1343051864" a:has_referenceConcept= "testOntology#Tuteur" a:inOIModel="testOntology" a:version="85"> ... <a:has_previousChange rdf:resource="#i-1524299176"/> </a:RemoveConcept> (3) <a:RemoveConcept rdf:ID="i-1343051887" a:has_referenceConcept= "testOntology#Animateur" a:inOIModel="testOntology" a:version="85"> ... <a:has_previousHistoryChange rdf:resource="#i-1343051864"/> </a:RemoveConcept> [...] (4) <a:AddConcept rdf:ID="i-1079545047999-24213731" a:has_referenceConcept="testOntology#FacilitateurApp" a:inOIModel="testOntology" a:version="85"> ... <a:has_previousChange rdf:resource="#i-1343051864"/> </a:AddConcept> (5) <a:AddSubConcept rdf:ID="i-1079545047999-24213758" a:has_referenceSubConcept="testOntology#AnimateurForum" a:has_referenceSuperConcept="testOntology#FacilitateurApp" a:inOIModel="testOntology" a:version="85"> ... </a:AddSubConcept>	(1) La classe *Professeur* a été effacée dans la version *85* de l'ontologie *testOntology*. (2) La classe *Tuteur* a été effacée dans la version *85* de l'ontologie *testOntology*. (3) La classe *Animateur* a été effacée dans la version *85* de l'ontologie *testOntology*. Le changement est induit par le changement (2). (4) La classe *FacilitateurApprentissage* a été ajoutée dans la version *85* de l'ontologie *testOntology*. (5) La classe *AnimateurForum* a été ajoutée dans la version *85* de l'ontologie *testOntology* comme sous-classe de la classe *FacilitateurApprentissage*.	Les classes *Professeur* et *Tuteur* ont été fusionnées dans une nouvelle classe, nommée *FacilitateurApprentissage*. À la classe résultante de la fusion lui a été transférée la classe *Animateur* (qui était avant une sous-classe de *Tuteur*) après lui avoir changé le nom en *AnimateurForum*.

La deuxième limitation concerne le langage utilisé pour déclarer les changements dans le fichier-journal du KAON : un langage interne à l'outil, qui n'est pas standardisé, non plus uniformisé pour considérer un formalisme de représentation des ontologies, comme OWL, par exemple. Ceci ne permet à aucun autre agent logiciel d'interpréter les fichiers-journal de KAON et de raisonner dessus puisque leur signification ne peut être interprétée en dehors de l'outil.

Enfin, la troisième limitation du processus de journalisation supporté par KAON concerne l'archivage des fichiers-journal, qui sont stockés localement, séparément des ontologies de référence. Ceci nécessite alors des mécanismes d'identification et d'accès aux fichiers-journal distribués afin de permettre à tout agent logiciel de récupérer des informations sur l'évolution des ontologies.

4.1.1.2.2. *Autres outils de journalisation de changements*

La plupart des éditeurs d'ontologies permettent l'enregistrement des opérations accomplies par les utilisateurs lors de la modification d'une ontologie (OntoWeb, 2002b). Cependant, ces opérations ne sont : (1) ni classifiées en fonction d'un modèle de changements ; (2) ni documentées complètement, en précisant, par exemple, les arguments d'entrée et/ou de sortie de toute opération accomplie lors du processus d'évolution ; (3) ni représentées dans un langage ayant une sémantique formelle normalisée ; (4) ni même stockées dans des archives spécifiques pour préserver la trace de l'évolution. Un exemple de ce type d'enregistrement est fourni par l'éditeur PROTÉGÉ, comme illustré dans la Figure IV-1.

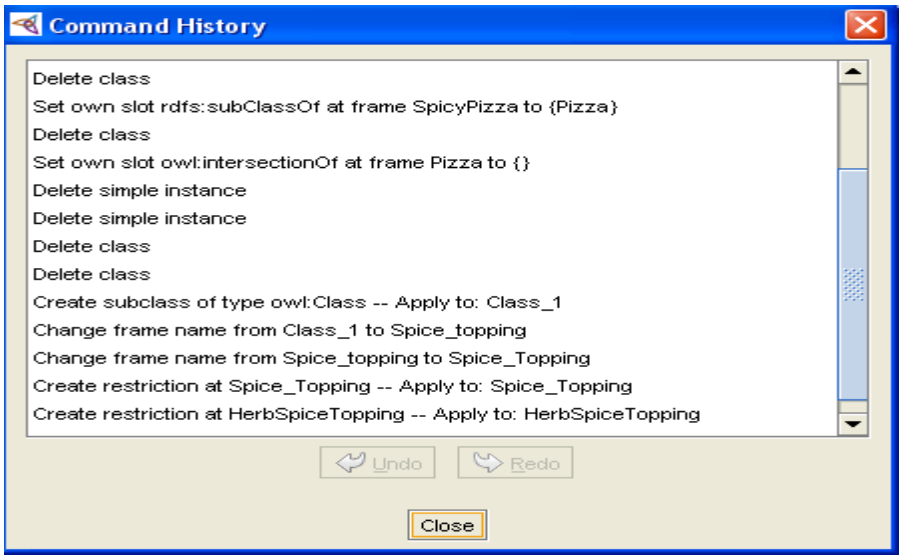


Figure IV-1. Journalisation de changements élémentaires dans PROTÉGÉ

Ainsi, bien que PROTÉGÉ peut fournir un historique des opérations élémentaires, celui-ci est loin d'être bien explicité puisque : les arguments d'entrée et de sortie des changements ne sont pas spécifiés (p.ex. l'énoncé « Delete_Class » ne nous dit pas de quelle classe précise il s'agit); les changements ne sont pas organisés pour transmettre la sémantique de l'évolution, comme la relation de causalité entre des changements primaires et additionnels par exemple, mais simplement listés d'une manière linéaire. De plus, cet historique n'est pas archivé en vue d'une récupération future ; il est simplement effacé une fois le processus d'évolution finalisé.

4.1.2. **Comparaison de versions d'ontologie**

Dans le contexte décentralisé et distribué du web, il est difficile d'avoir accès en tout temps aux fichiers-journal des changements. Il est également difficile d'interpréter ces fichiers, étant donné qu'ils sont explicités dans un langage interne à la machine (Noy, 2004). Pour résoudre ce problème, la deuxième approche repose sur l'identification des changements par une comparaison entre deux versions d'ontologie. Cette approche est supportée actuellement par deux outils – **OntoView** et **PromptDiff** – que nous discutons dans les paragraphes suivants.

4.1.2.1. **L'outil OntoView**

Pour identifier des changements ontologiques, OntoView (Klein, 2004; Klein, Fensel, Kiryakov, et Ognyanov, 2002) compare les versions d'une ontologie au niveau structural, c'est-à-dire au niveau des définitions RDF de classes et propriétés, le but étant de découvrir les définitions ayant été modifiées d'une version d'ontologie à une autre (*cf.* Figure IV-2).

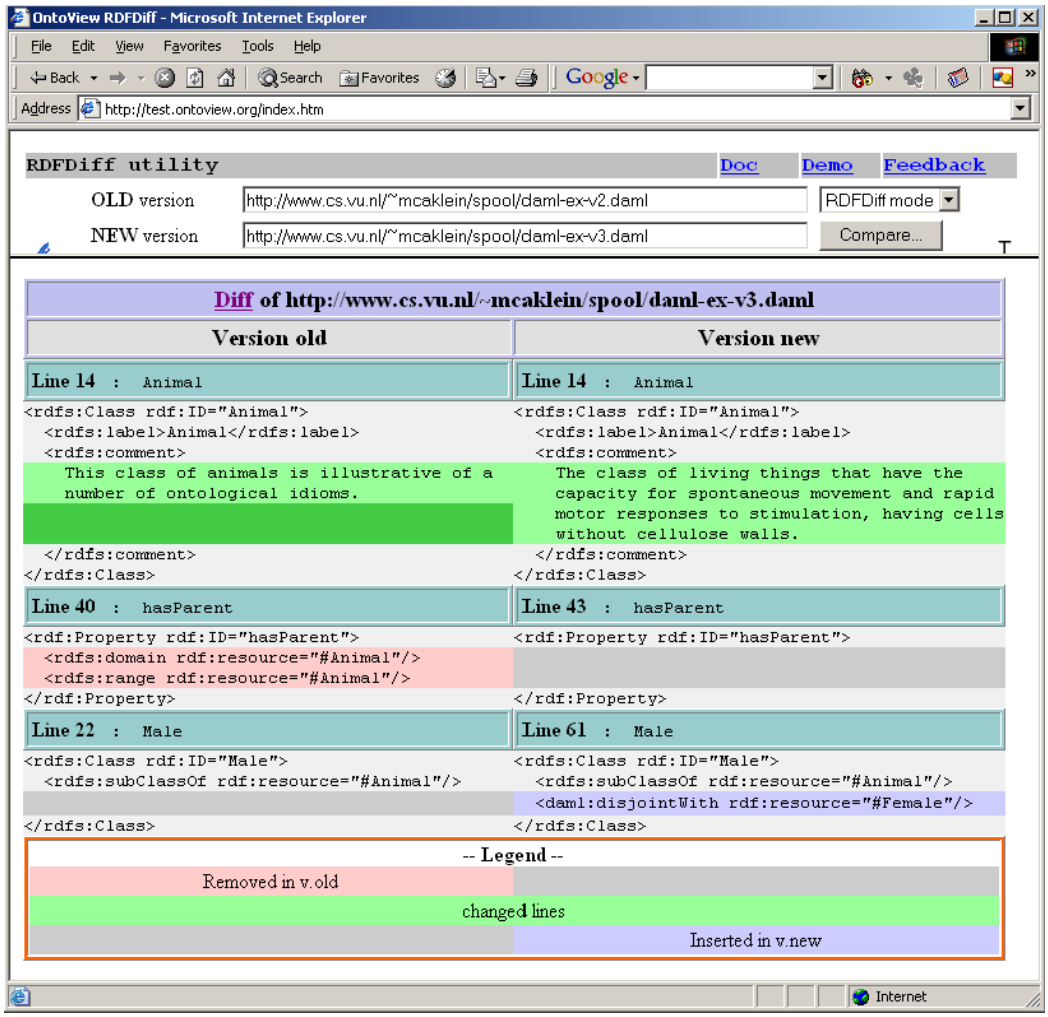


Figure IV-2. Comparaison effectuée par OntoView
Comparaison effectuée entre les définitions RDF de classes et propriétés appartenant à deux versions différentes pour identifier ce qui a été modifié (lignes soulignées en vert), ce qui a été effacé (lignes en rouge) et ce qui a été ajouté (lignes en bleu).

Pour chaque définition modifiée, OntoView identifie ensuite ce qui a été effacé, ajouté ou transformé à l'intérieur de cette définition, en mettant en évidence ces informations à l'aide d'un code de couleur.

Même si l'outil peut identifier un certain nombre de changements, il ne peut pas préciser si la signification d'une classe a été vraiment modifiée ou seulement explicitée dans la nouvelle version d'ontologie. C'est pourquoi les auteurs affirment qu'une extension d'OntoView devrait permettre aux utilisateurs de caractériser manuellement l'effet d'un changement sur la conceptualisation d'une classe en termes de « identique », si la conceptualisation n'a pas été modifiée, ou « conceptuellement différente », si la conceptualisation a été modifiée.

4.1.2.2. **L'outil PromptDiff**

PromptDiff (Noy, 2004; Noy et Musen, 2003b), tout comme OntoView, effectue une comparaison des versions d'ontologie. Néanmoins, alors qu'OntoView se focalise sur l'identification des différences structurales entre deux versions d'ontologie, PromptDiff se focalise sur l'identification des similarités en utilisant un algorithme de mise en correspondance des versions (Noy et Musen, 2002). Cet algorithme[35], comme l'illustre la Figure IV-3, mette en correspondance deux versions différentes d'une même ontologie pour identifier si les classes et les propriétés restent *isomorphiques* ou sont *changées*. En soulignant les classes pour lesquelles aucune mise en correspondances n'a été trouvée, l'algorithme de PromptDiff est également capable d'identifier les classes ajoutées à une version ou celles effacées d'une version.

[35] Par exemple, un tel algorithme mettrait en correspondance une hiérarchie de classes appartenant à une version avec la même hiérarchie, mais d'une autre version, pour mettre en évidence un changement ayant modifié le nom de la classe racine de cette hiérarchie.

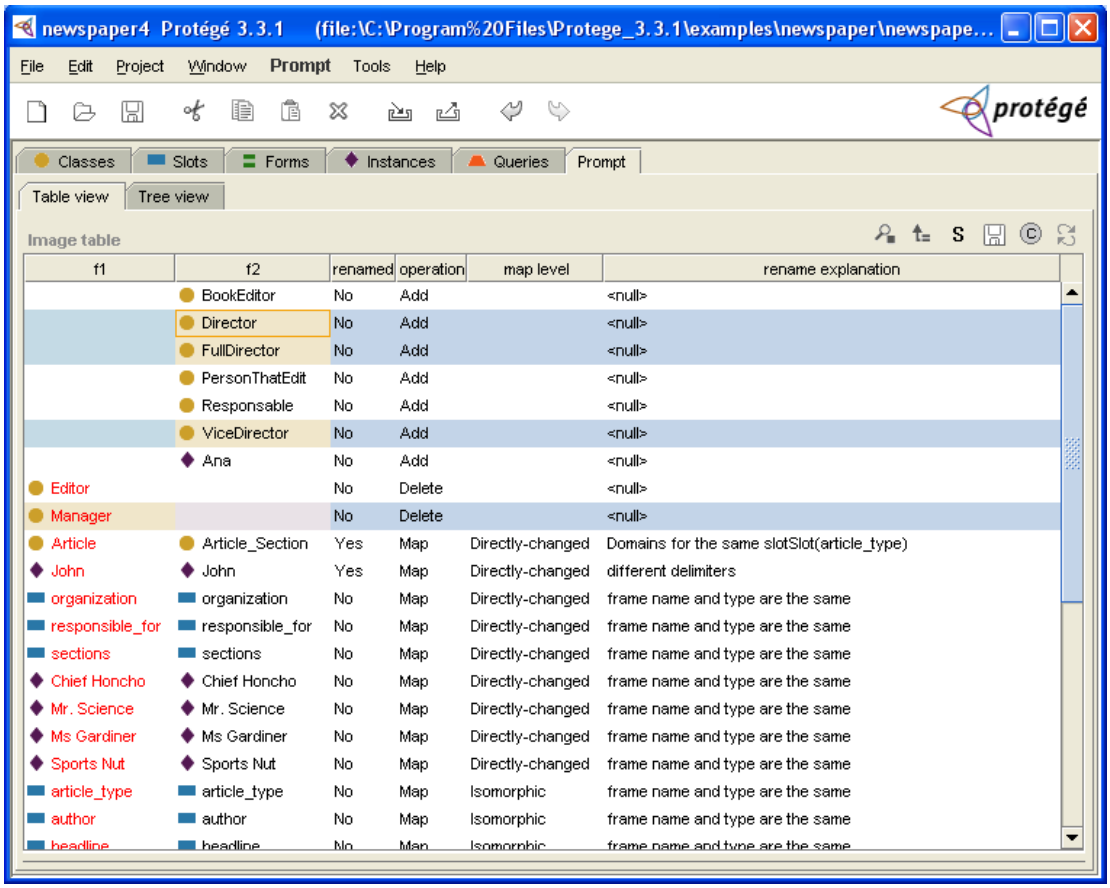


Figure IV-3. Comparaison effectuée par PromptDiff

Comparaison effectuée entre deux versions différentes (f1 et f2) pour identifier les classes, les propriétés et les instances qui ont été ajoutées, effacées ou celles dont l'isomorphisme a été préservé

4.1.2.3. **Discussion des outils OntoView et PromptDiff**

Les deux outils – **OntoView** et **PromptDiff** – présentent l'avantage d'être capables d'identifier des changements sans autre information que celle fournie par les versions d'ontologie et par les algorithmes de mise en correspondance des versions. Ils présentent toutefois un certain nombre de limitations.

Premièrement, les outils ne sont pas toujours capables d'interpréter avec précision les mises en correspondance effectuées, ni d'extraire les « vrais » changements apportés[36] aux versions d'ontologie. Ou, même s'ils identifient un certain nombre de changements, ceux-ci sont uniquement de type additif ou soustractif, les changements plus complexes étant, pour le moment, en dehors de la portée des outils de comparaison des versions d'ontologie. Ceci parce que les algorithmes utilisés ne peuvent pas rendre compte de la sémantique extrêmement riche de l'ensemble des changements possibles dans une ontologie.

La deuxième limitation réfère à l'impossibilité d'identifier l'ordre d'implémentation de changements et d'établir des relations de causalités entre les changements. Ainsi, ni OntoView ni PromptDiff ne peuvent considérer les stratégies d'évolution utilisées, ce qui entraîne une perte importante de données portant sur la signification du processus d'évolution et les choix effectués durant ce processus.

Enfin, la troisième limitation[37] concerne le fait que la comparaison fondée sur des algorithmes d'identification des changements, comme c'est le cas d'OntoView, ou sur des heuristiques de mises en correspondances des versions d'ontologie, comme c'est le cas du PromptDiff, est une grande consommatrice de temps et de ressources. Dans la vision du Web sémantique, où des agents en communication ont besoin d'un accès rapide et facile à toute information afin de répondre aux requêtes ponctuelles des utilisateurs (Hendler, 2001), ce type de comparaison pourrait s'avérer assez contraignant.

4.1.3. **Journalisation de changements ou comparaison des versions : discussion**

Quelle approche d'identification des changements choisir, sachant que l'une et l'autre possèdent des qualités d'une importance majeure pour le Web sémantique ? Dans cette section nous discutons des avantages et des limites de chacune des approches en nous situant

---

[36] Par exemple, si on modifie le nom d'une classe (de `Manager` en `Director`, *cf.* Figure IV-3), et on lui ajoute deux sous-classes et une propriété, aucun de deux outils ne peut effectuer une mise en correspondance pour ressortir qu'il s'agit d'une même classe. Au contraire, ils décrivent cette situation comme étant l'effacement de la classe `Manager` et l'ajout d'une nouvelle classe, `Director`.

[37] Précisons toutefois que cette limitation dépend de la taille de versions d'ontologies à comparer (des versions de grande ou petite taille).

dans la perspective du *Web sémantique comme architecture décentralisée et distribuée* et ensuite dans la perspective du *Web sémantique comme architecture fondée sur le référencement sémantique des ressources*.

4.1.3.1. **Discussion des approches selon la perspective du Web sémantique comme architecture décentralisée et distribuée**

Considérons d'abord la perspective du *Web sémantique comme architecture décentralisée et distribuée*, une architecture où les agents sont autonomes et non dirigés, où les ontologies et leurs versions sont distribuées et non stockées dans une libraire unique et prédéfinie. Si on se situe dans cette perspective, l'approche par journalisation présente un certain nombre des désavantages :

- Problème d'accès aux fichiers-journal étant donné que, la plupart du temps, ils sont stockés localement, dans des archives propres aux outils de journalisation les ayant produits. À ce problème d'accès peut s'ajouter aussi celui d'identification du fichier-journal, correspondant à un couple de versions (V_N, V_{N+1}), parmi tous les fichiers existants à un moment donné.

- Problème d'interprétation des fichiers-journal dû, d'une part, au fait que ces fichiers sont formalisés dans un langage machine spécifique et d'autre part, au fait qu'il n'y a actuellement aucun modèle standard de représentation des changements.

L'approche par comparaison de versions, contrairement à celui de journalisation, présente un avantage majeur dans le contexte du Web décentralisé et distribué, à savoir le fait qu'elle est capable d'identifier des changements en utilisant uniquement des versions d'ontologie ainsi qu'un ou plusieurs algorithmes prédéfinis pour mettre en correspondance ces versions. Cette approche n'a donc besoin d'aucune autre source information supplémentaire, évitant ainsi les problèmes dus à la localisation et à l'accès aux fichiers-journal.

4.1.3.2. **Discussion des approches selon la perspective du Web sémantique comme architecture fondée sur le référencement sémantique des ressources**

Considérons maintenant la perspective du *Web sémantique comme architecture fondée sur le référencement sémantique des ressources* Web, une architecture où des processus de « recherche intelligente », fondés sur des ontologies, peuvent décharger les utilisateurs d'une bonne partie de leur tâche de recherche, de construction et de combinaison des résultats grâce aux capacités accrues des machines à accéder aux contenus des ressources et à effectuer des raisonnements sur ceux-ci. Dans cette perspective, qui est à la base du vrai Web sémantique, le principal problème réside dans le fait que l'évolution des ontologies peut produire un effet très problématique sur l'intégrité de l'accès aux ressources référencées ou sur leur interprétation, comme nous le montrons dans le chapitre V. Pour résoudre ce problème, le plus important est d'avoir une vue sur la totalité des changements apportés aux ontologies. De ce point de vue, l'approche par journalisation présente des avantages bien plus intéressants que celle par comparaison :

- Les changements, qu'ils soient élémentaires ou complexes, peuvent être récupérés intégralement et de façon non ambiguë lors du processus d'évolution. De ce fait, on peut avoir une vision complète du processus d'évolution et de la signification exacte de chacun des changements.

- Les changements sont représentés formellement dans des fichiers-journal et peuvent donc être traités formellement en vue d'obtenir une analyse des effets de changements sur le référencement sémantique de ressources.

Comparer deux versions d'ontologie peut aussi nous fournir des informations sur les changements accomplis pour passer d'une version à une autre. Néanmoins, l'approche par comparaison peut identifier uniquement des changements de type additif ou soustractif, ce qui ne garantit pas l'intégralité de changements ontologiques apportés lors d'un processus d'évolution. Cette approche ne garantit ni la véracité des changements identifiés, ni de quelle manière les changements sont reliés l'un à l'autre. Concernant l'identification des changements plus complexes, il n'existe actuellement aucun algorithme capable de les traiter. Bien que Noy, Kunnatur, Klein et Musen (2003) se sont dernièrement attardés à enrichir

l'algorithme utilisé par PromptDiff avec un ensemble de *matchers* permettant l'identification, en plus de changements additifs ou soustractifs, de déplacements des classes ou des instances, les auteurs soulignent néanmoins des problèmes d'efficience du fait que l'algorithme est très compliqué[38].

4.1.3.3. **Conclusion et choix de l'approche par journalisation**

Pour résumer, en fonction de la perspective du web sémantique dans laquelle nous nous situons, une approche d'identification des changements peut être plus pertinente que l'autre. Bien que l'approche par comparaison peut résoudre les problèmes d'accès et d'interprétation des fichiers-journal distribués, elle ne nous garantit ni la véracité ni l'intégralité de l'ensemble des changements apportés pour passer d'une version d'ontologie à une autre. À cause de ces limitations, dans cette thèse nous nous somme concentrée sur l'approche par journalisation. Le facteur nous ayant influencée est le suivant. Notre contexte de recherche concerne le Web sémantique éducatif dont l'architecture est fondée sur l'association de références sémantiques à des ressources distribuées afin de décrire formellement leur contenu. Plus spécifiquement, les systèmes TELOS et ERPAD (*cf.* paragraphe 1.2.2.) sont des applications dont l'architecture est fondée sur le référencement sémantique des ressources à l'aide des ontologies évolutives.

Une difficulté pointe à l'horizon : l'évolution des ontologies peut produire des conséquences dramatiques sur l'accès aux ressources référencées ou sur leur interprétation. Pour traiter ces conséquences, une vue complète des changements, une vue qu'illustre avec précision la signification réelle des changements ontologiques, une vue qui présente non seulement les changements, mais aussi le lien de causalité qui les relie (c.-à-d. changement primaire *vs* additionnel), devient plus qu'intéressante, elle devient essentielle. C'est pourquoi l'approche par journalisation est, dans notre cas, plus appropriée que celle par comparaison.

---

[38] L'algorithme utilise un ensemble de *matchers* pour mettre en correspondance les classes appartenant à V_N avec celles appartenant à V_{N+1}. Ensuite, pour chaque correspondance trouvée – i.e. classe appartenant à V_N et à V_{N+1} – l'algorithme regarde si ses sous-classes en V_N sont identiques avec celles en V_{N+1}. Sinon, alors la classe en question a été déplacée. Ce procédé est néanmoins extrêmement coûteux, c'est pourquoi les auteurs ont adopté la solution de ne l'appliquer que pour la partie de l'ontologie qui est sélectionnée par les utilisateurs.

Dans la section suivante, nous allons présenter l'approche par journalisation que nous proposons pour l'évolution des ontologies, ainsi que le système **ChangeHistoryBuilder** qui l'implémente. Nous anticipons la présentation du système, en soulignant que nous n'avons pas eu l'intention de développer un autre modèle de journalisation, mais bien un modèle qui s'inspire des avantages d'approches existantes et qui résout, en même temps, un certain nombre de leurs désavantages, comme le fait de considérer les changements complexes, en plus de ceux élémentaires, ou de permettre une représentation des changements la plus standardisée possible. Nous avons trouvé également un moyen d'intégrer le fichier-journal directement dans la nouvelle version d'ontologie, afin d'éviter les problèmes de localisation et d'accès aux fichiers-journal.

4.2 **Journalisation des changements avec *ChangeHistoryBuilder* (CHB)**

Dans cette section, nous présentons les deux fonctionnalités de base de **ChangeHistoryBuilder – CHB** (*cf.* Figure IV-4), un de deux modules qui composent le cadre de référence pour la gestion des changements apportés aux versions d'ontologie, cadre qui est le sujet de cette thèse.

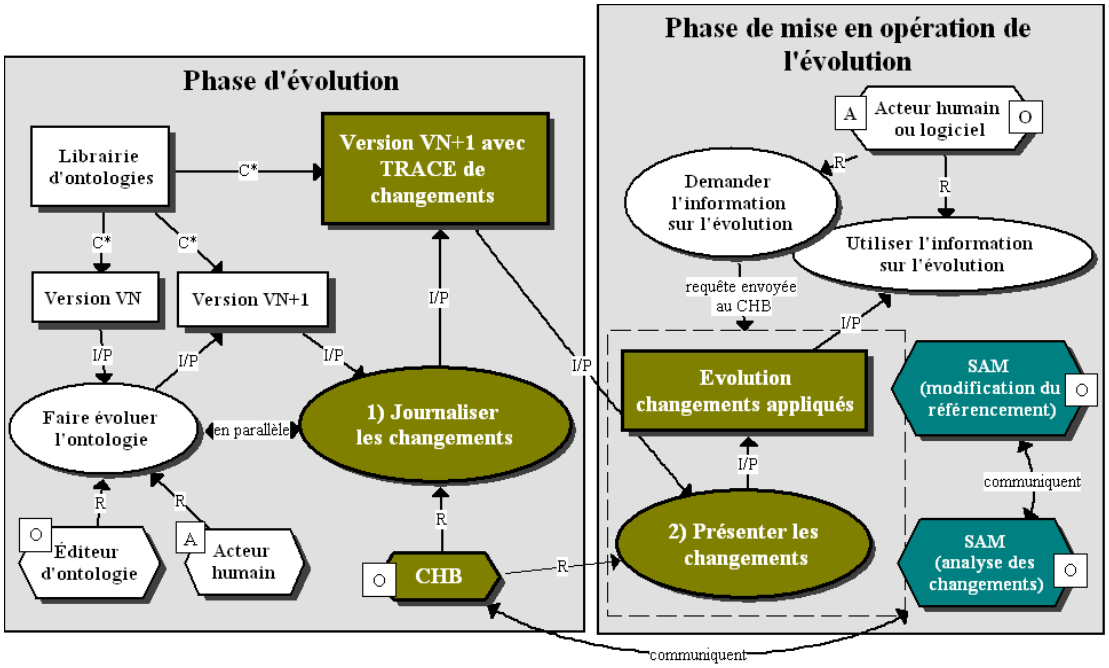


Figure IV-4 Les deux fonctionnalités du système CHB
(La légende du formalisme graphique standard du MOT est consultable dans l'Appendice A)

La **première fonctionnalité** du système CHB est celle de **journaliser les changements** durant la phase d'évolution d'une ontologie. Le CHB crée un historique des changements effectués lors de l'évolution en utilisant : (1) un modèle des métadonnées pour la journalisation standardisée des changements ; (2) un langage de formalisation des changements ayant une sémantique formelle riche et consistante, permettant aux agents logiciels d'effectuer des raisonnements complexes. Cette fonctionnalité est présentée dans la Section 4.2.1.

La **deuxième fonctionnalité** consiste à récupérer la trace de changements formalisés, d'extraire les changements exécutés lors de l'évolution et de les présenter à l'utilisateur. Précisons que, dans cette thèse, nous avons abordé l'identification de changements effectués pour passer de V_N à V_{N+1} et non pas à V_{N+i} (où i est un entier supérieur à 1). Cette fonctionnalité est présentée dans la Section 4.2.2.

Le système CHB a été développé selon une approche de programmation orientée objet avec le langage Java. Ce choix se justifie par le fait qu'il permet de définir des objets et des fonctions standard et de concevoir des modules informatiques fonctionnant d'une manière indépendante. Comme il s'agit d'un prototype exploratoire, nous nous sommes davantage focalisée sur la conceptualisation sous-jacente au système que sur son implémentation. Ainsi, bien que nous ayons fait la preuve que les notions et les solutions proposées sont soutenables informatiquement, nous n'avons cependant pas développé des programmes complexes pour les fonctions ou pour l'interface. C'est pourquoi nous ne donnons pas de détails sur l'implémentation du CHB, autres que les interactions et les résultats fournis par le système.

4.2.1. **Journalisation des changements supportée par le système CHB**

L'approche par journalisation possède l'avantage de tracer la sémantique des changements apportés aux ontologies. Elle présente cependant plusieurs désavantages reliés à l'accès aux fichiers-journal, à l'interprétation de ces fichiers par des outils hétérogènes ainsi qu'à la manière de capturer l'intention exacte de changements. C'est autour de ces questionnements que nous avons ancré la conception du CHB, à savoir :

- Y a-t-il une possibilité de rendre le processus de journalisation plus standardisé, afin de résoudre les problèmes d'interprétation des fichiers-journal trop orientés vers un langage interne à la machine ?

- Y a-t-il une possibilité de représenter ces fichiers-journal dans un langage ayant une sémantique formelle riche et consistante ? Ce langage ne pourrait-il pas s'approcher le plus possible d'un langage déjà standard, soit OWL ?

- Y a-t-il une possibilité de rendre ces fichiers-journal plus facilement accessibles en les attachant, par exemple, aux versions de l'ontologie ? Par ce moyen, n'évitons-nous pas les problèmes d'accès aux fichiers-journal distribués, puisque pour identifier les changements, nous avons uniquement des versions d'une ontologie ? Ceci ne rejoint-il pas le principal avantage de l'approche par comparaison ?

- Finalement, est-il possible de rendre compte de la richesse sémantique sous-jacente à tout processus d'évolution en permettant la journalisation des changements complexes, en plus des changements élémentaires ?

4.2.1.1. **Étapes générales de la journalisation des changements**

Nous répondons à ces questions en proposant une démarche de journalisation des changements en trois étapes, tel qu'illustré dans la Figure IV-5. Cette démarche est supportée par le système CHB.

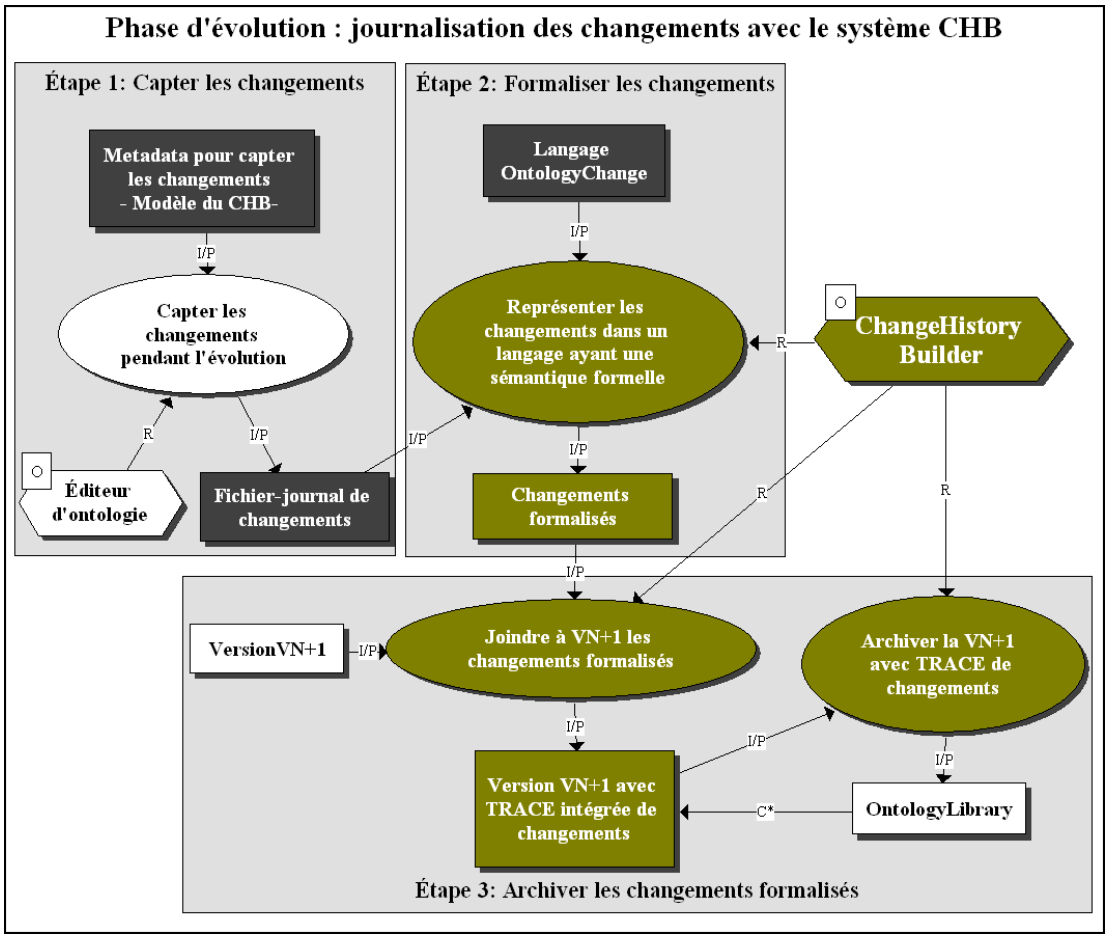


Figure IV-5. Journalisation par étapes du système CHB
(La légende du formalisme graphique standard du MOT est consultable dans l'Appendice A)

La première étape – **Capter les changements** – vise à enregistrer dans un fichier-journal les changements lors de l'évolution d'une ontologie. Cette étape, courante d'ailleurs

dans tout processus de journalisation, nécessite néanmoins une attention spéciale en ce qui concerne la richesse sémantique et la non-ambigüité des changements récupérés, mais aussi l'enregistrement de ces changements d'une manière qui pourrait résoudre les problèmes d'interprétation de fichiers-journal provenant de divers éditeurs. Cette étape est présentée dans la Section 4.2.1.2.

La deuxième étape – **Formaliser les changements** – concerne la représentation des changements dans un langage, que nous nommons *OntologyChange*, ayant une sémantique formelle riche et consistante puisqu'elle est fondée sur la sémantique OWL. Ces changements représentés formellement permettront aux agents logiciels d'extraire des informations relatives à l'évolution d'une ontologie et d'appliquer des raisonnements sur ces informations pour identifier les effets de changements sur le référencement sémantique de ressources. Cette étape est présentée dans la Section 4.2.1.3.

La troisième étape – **Archiver les changements formalisés** - consiste dans l'archivage de changements formalisés à l'étape précédente. C'est ici qu'on peut pallier l'avantage d'avoir accès à un fichier-journal qui capture la sémantique du processus d'évolution avec celui d'identifier les changements en disposant uniquement des versions d'une ontologie. La solution que nous avançons est celle de joindre les changements formalisés, apportés à V_N pour obtenir V_{N+1}, directement à la version V_{N+1}. Nous obtenons ainsi la nouvelle version de l'ontologie avec une trace intégrée de changements. Nous nommons cette version $\mathbf{V_{N+1}Change}$. Nous présentons cette étape dans la Section 4.2.1.4.

4.2.1.2. **Capter les changements avec le modèle du CHB (étape 1)**

Cette étape vise à enregistrer dans le fichier-journal les changements effectués lors de l'évolution d'une ontologie. Dans les trois sections ci-dessous, nous présentons le rôle du CHB, puis son modèle de journalisation et finalement, une discussion du modèle.

4.2.1.2.1. *Rôle du CHB dans la capture des changements*

Pour résoudre les problèmes d'interprétation des fichiers-journal provenant de divers éditeurs d'ontologies[39], il faudrait offrir à ces éditeurs un **modèle commun de journalisation des changements**. C'est la réponse apportée par le modèle du CHB que nous proposons pour supporter les éditeurs d'ontologies à capter et à enregistrer le processus d'évolution d'une manière plus normalisée et plus riche sémantiquement. Ce modèle, tout en étant assez flexible pour s'accommoder aux comportements de différents éditeurs, doit proposer une vue normalisée de types de changements et de données à récupérer pour documenter ces changements. De cette manière, les éditeurs peuvent intégrer le modèle du CHB directement dans le processus d'édition des changements et produire des fichiers-journal présentant une description, normalisée et riche, des changements effectués.

La Figure IV-6 présente le rôle du CHB dans l'étape de capture des changements.

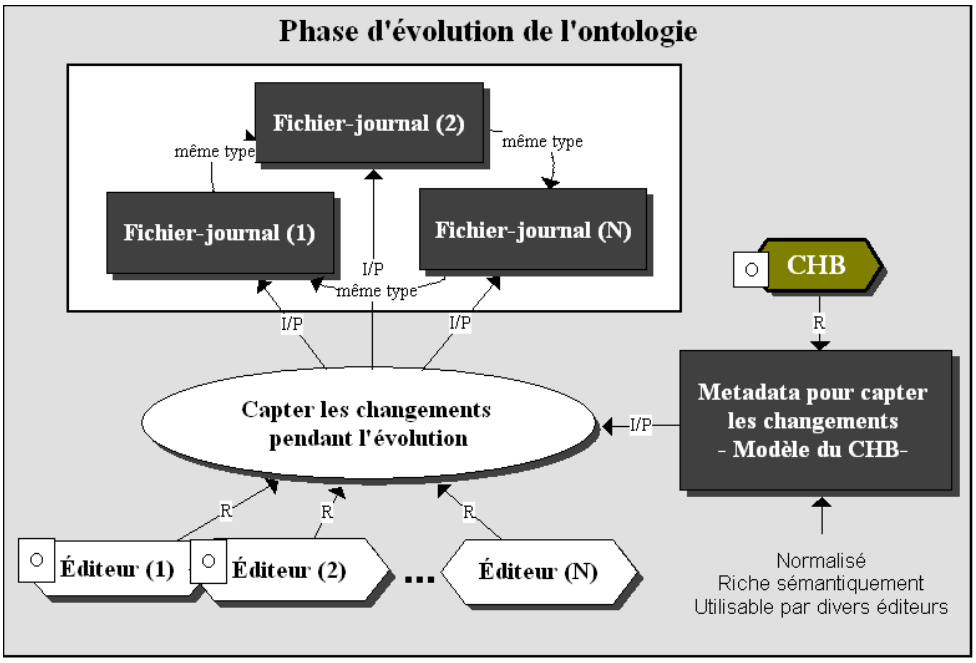


Figure IV-6. Rôle du CHB dans l'étape de capture des changements
(La légende du formalisme graphique standard du MOT est consultable dans l'Appendice A)

[39] Même si les éditeurs d'ontologies possèdent une fonctionnalité qui leur permet de journaliser des changements, ces derniers sont récupérés d'une manière peu organisée et sont représentés dans un langage interne à l'outil.

Observons que nous avons délégué aux éditeurs d'ontologies la responsabilité d'intégrer le modèle du CHB. Nous avons ainsi choisi de cibler le développement d'un modèle conceptuel de journalisation, plutôt que l'implémentation effective de ce modèle dans un éditeur actuel. Les raisons de notre choix sont multiples : (a) concevoir un modèle applicable à plusieurs éditeurs, sans subir l'influence d'un éditeur précis ; (b) spécifier les aspects nécessaires à une journalisation des changements qui rende explicite la richesse sémantique de tout processus d'évolution, comme les relations de causalité entre les changements ou encore la complexité des changements applicables ; (c) proposer un modèle conceptuel de journalisation et faire la preuve du concept.

4.2.1.2.2. *Modèle de journalisation du CHB*

Nous présentons dans le Tableau IV-3, ainsi que dans le Tableau IV-4, l'ensemble des métadonnées du modèle du CHB. Précisons que nous avons spécifié ce modèle en fonction des ontologies OWL.

Tableau IV-3. Modèle de journalisation du CHB : information sur le processus de journalisation

Metadata	Signification	Valeur
Un fichier-journal de type *CHB* se compose d'un entête pour déclarer des informations à propos du processus de journalisation.		
<ChangeLogInfo>	Élément pour déclarer des informations sur le fichier-journal.	Cet élément est une simple balise.
<VersionOld>	Élément pour déclarer la version V_N de l'ontologie	Ces deux éléments prennent comme valeurs respectivement l'URI (*Uniform Resource Identifier*) de la version V_N et V_{N+1}.
<VersionNew>	Élément pour déclarer la version V_{N+1} de l'ontologie	
<Auteur >	Élément pour déclarer l'auteur du processus d'évolution	Valeur de type texte ou autre formalisme.

Tableau IV-4. Modèle de journalisation du CHB : information sur les changements

Metadata	Signification	Valeur
Un fichier-journal de type *CHB* se compose aussi des déclarations de changements et de caractéristiques de changement.		
<ChangeList>	Éléments d'entête pour la description d'une liste de changements.	Aucune valeur. Ils sont juste des balises : (1) <ChangeList>, pour délimiter le bloc de déclaration des changements, et (2) <Change> pour délimiter le bloc de déclaration d'un changement spécifique.
<Change>	Élément d'entête pour la description de chaque changement à l'intérieur de la liste de changements.	
<ChangeNumber>	Élément pour déclarer une relation de précédence entre les changements exécutés pour passer de V_N à V_{N+1}.	Nombres entiers positifs (*cf.* les propriétés d'objet `haveNumber` et `haveParentNumber` spécifiées dans l'ontologie des changements). *Rémarque :* Si la valeur zéro est enregistrée pour <ChangeParentNumber>, alors le changement identifié par <ChangeNumber> est un changement primaire. Pour toute autre valeur enregistrée pour <ChangeParentNumber>, le changement identifié par <ChangeNumber> est un changement additionnel.
<ChangeParentNumber>	Élément pour déclarer une relation de causalité en précisant l'identifiant numérique du changement-parent du changement additionnel identifié par <ChangeNumber>. *Remarque* : Ce changement-parent n'est pas nécessairement un changement primaire. Il peut être un changement additionnel qui engendre d'autres changements additionnels.	

<ChangeType>	Élément pour spécifier le type général d'un changement (p.ex. ajout ou fusion). *Remarque* : Cet élément est optionnel. Cette information peut être extraite à partir de <ChangeOperation>.	Une de classes de premier niveau de la hiérarchie des changements élémentaires et complexes (*cf.* Section 3.2).
<ChangeOrder>	Élément pour préciser si le changement est primaire ou additionnel. *Remarque* : Cet élément est optionnel. Cette information peut être extraite à partir de <ChangeParentNumber>.	Une et seulement une de deux chaînes littérales suivantes : ('Primaire' ; 'Additionnel').
<ChangeOperation>	Élément pour décrire le changement spécifique (p.ex. ajout ou fusion de classes ou de propriétés).	Une de classes du dernier niveau de la hiérarchie de changements élémentaires et complexes (*cf.* Section 3.2).
<ArgumentX>	L'élément <ChangeOperation> spécifie une transformation dont le domaine est une liste de variables appartenant au V_N (la liste des <ArgumentX>) et le codomaine est une liste de variables appartenant à V_{N+1} (la liste des <ArgumentY>). *Remarque* : Pour un changement de type « *Add* » la liste des <ArgumentX> est vide. Pour le « *Delete* », la liste des <ArgumentY> est vide.	Un ou plusieurs ID (l'identifiant unique) de classes ou de propriétés déclarées en V_N et en V_{N+1}.
<ArgumentY>		
<Comments>	Élément pour permettre la déclaration des commentaires.	Texte en langage naturel ou autres formalismes.

Une description d'une opération de changement, la `<ChangeOperation>`, peut contenir un ou plusieurs éléments de type `<ArgumentX>` et `<ArgumentY>`. Pour préciser la signification de la relation qui existe entre les valeurs des éléments, ces derniers doivent suivre un ordonnancement prédéfini en fonction de chaque changement. Dans le Tableau IV-5 et le Tableau IV-6 nous présentons quelques exemples qui montrent comment l'ordonnancement des `<ArgumentX>` et `<ArgumentY>` explicite une signification particulière en fonction de l'opération de changement associée (d'autres exemples peuvent être consultés dans l'Appendice D).

Tableau IV-5 Signification de l'ordonnancement des <ArgumentX> et <ArgumentY> par rapport aux changements de classes

<ChangeOperation>	Signification de l'ordonnancement des <ArgumentX>	Signification de l'ordonnancement des <ArgumentY>
<DeleteClass>	Couple de valeurs [$ArgumentX_1$, $ArgumentX_2$], où : - $ArgumentX_1$ = l'ID de la classe effacée; - $ArgumentX_2$ = l'ID de la superclasse de la classe effacée.	La liste des <ArgumentY> est vide.
<MoveDisjointClass>	Couple de valeurs [$ArgumentX_1$, $ArgumentX_2$], où : - $ArgumentX_1$ = l'ID de la classe qui contient l'axiome de disjonction en V_N; - $ArgumentX_2$ = l'ID de la classe étant déclarée disjointe avec celle précisée par $ArgumentX_1$.	Couple de valeurs [$ArgumentY_1$, $ArgumentY_2$], où : - $ArgumentY_1$ = l'ID de la classe où a été transféré l'axiome de disjonction en V_{N+1}; - $ArgumentY_2$ = même valeur que celle de l'$ArgumentX_2$.
<MergeClasses>	Liste [$ArgumentX_1$ … $ArgumentX_N$] où chaque argument est l'identifiant d'une des classes fusionnées.	Couple de valeurs [$ArgumentY_1$, $ArgumentY_2$], où : - $ArgumentY_1$ = l'ID de la classe résultant de la fusion; - $ArgumentY_2$ = l'ID de la superclasse de la classe résultante.

Tableau IV-6 Signification de l'ordonnancement des <ArgumentX> et <ArgumentY> par rapport aux changements de propriétés

<ChangeOperation>	Signification de l'ordonancement des <ArgumentX>	Signification de l'ordonancement des <ArgumentY>
<AddPropertyDomain>	La liste des <ArgumentX> est vide.	Couple de valeurs [$ArgumentY_P$, $ArgumentYD_{1..N}$], où : - $ArgumentY_P$ = l'ID de la propriété dont le domaine a été élargi par l'ajout d'une ou plusieurs classes, suivi de l'indicatif *ObjectProperty* ou *DatatypeProperty*; - $ArgumentYD_{1..N}$ = liste composée de l'ID de chaque classe ajoutée au domaine de la propriété Y_P.
<ModifyPropertyRange>	Couple de valeurs [$ArgumentX_P$, $ArgumentX_R$], où : - $ArgumentX_P$ = l'ID de la propriété dont le range a été modifié. - $ArgumentX_R$ = l'ID de la classe, appartenant au range de la propriété, qui a été échangée avec une ou plusieurs classes.	Couple de valeurs [$ArgumentY_P$, $ArgumentYR_{1..N}$], où : - $ArgumentY_P$ = même valeur que celle de l'$ArgumentX_P$; - $ArgumentYR_{1 \ldots N}$ = liste composée de l'ID de chaque classe qui a été échangée contre X_R.

La Figure IV-7 illustre un exemple de fichier-journal où les changements sont spécifiés en utilisant les métadonnées fournies par le modèle de journalisation du CHB.

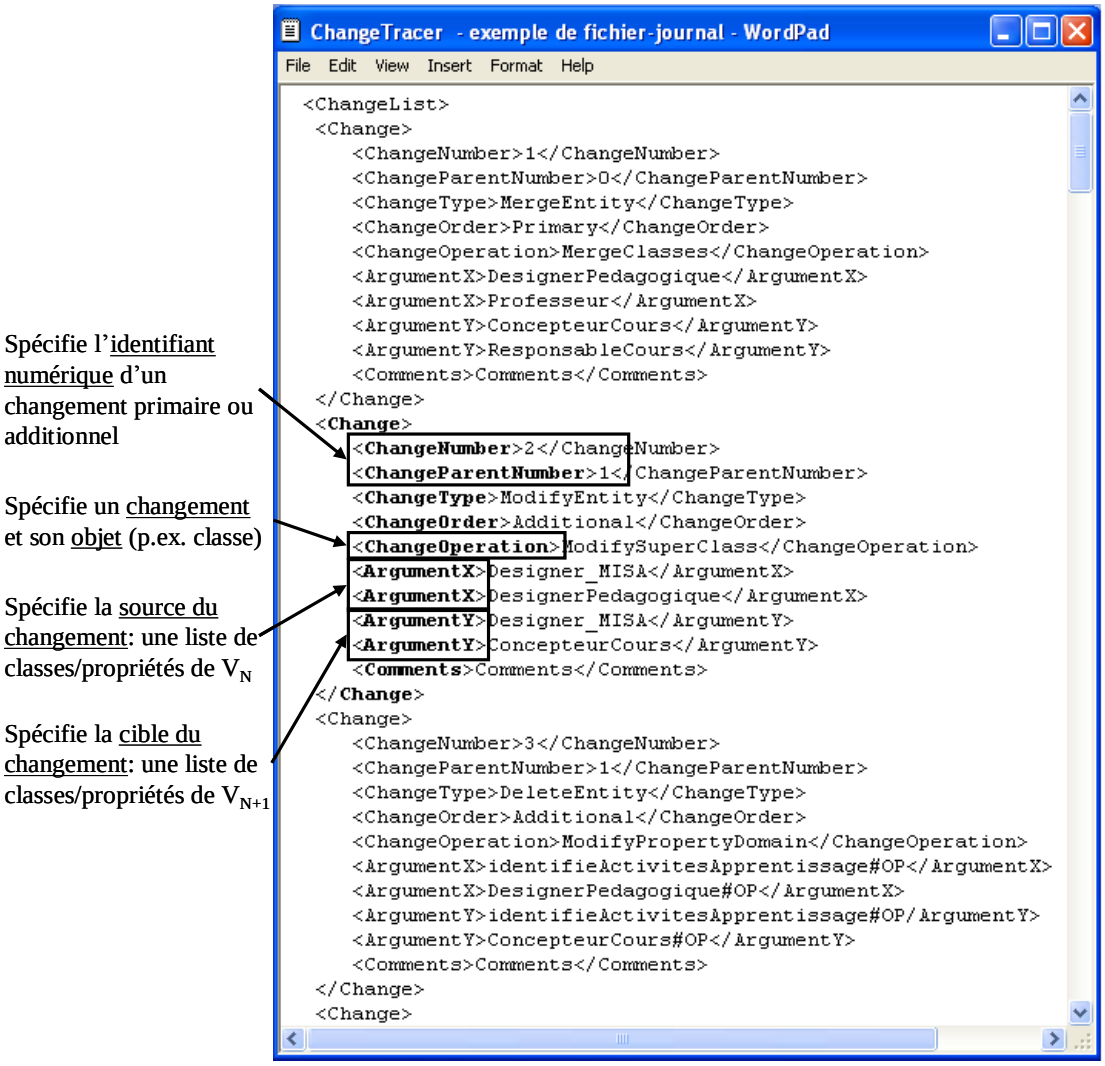


Figure IV-7. Exemple de fichier-journal obtenu avec le modèle du CHB

4.2.1.2.3. *Discussion du modèle du CHB*

Le modèle du CHB se situe dans la voie ouverte par Concordia, c'est-à-dire qu'il propose un modèle de description des changements pour une journalisation uniformisée et

cohérente. Toutefois, alors que le modèle Concordia propose un ensemble des schémas pour documenter les changements, chaque schéma étant syntaxiquement différent, le modèle du CHB propose un ensemble de métadonnées, rassemblées dans une structure unique est commune à tous les changements, bien que la signification d'éléments `<ArgumentX>` et `<ArgumentY>` puisse différer. Le fait d'avoir une structure plus générique peut faciliter l'utilisation du modèle du CHB par une diversité d'éditeurs d'ontologies, qui produiront ainsi des fichiers-journal dont l'interprétation est uniformisée.

Dans la mesure où le modèle du CHB permet la description des changements complexes et la description de la relation de causalité existant entre ceux-ci (c.-à-d. changement primaire/additionnel), il est plus riche sémantiquement que le modèle CONCORDIA. En le comparant cependant au modèle de journalisation supporté par KAON, le modèle du CHB présente une certaine ressemblance : les valeurs prises par les éléments `<ChangeType>` et `<ChangeOperation>` sont des classes provenant d'une ontologie de changements. Néanmoins si l'ontologie de KAON permet simplement la représentation des changements élémentaires, notre ontologie prend en compte des changements plus complexes, comme la fusion/division de classes ou la modification du domaine/codomaine de propriétés. Elle est, en plus, spécialisée pour décrire des changements applicables aux ontologies OWL.

Comme illustré dans la Figure IV-8, le fichier-journal de type CHB n'a pas une structure linéaire. Au contraire, il est organisé de façon que chaque changement primaire est représenté sous la forme d'un arbre formé par ses conséquences, c'est-à-dire par ses changements additionnels. Cet arbre des changements, loin d'être prédéfini pour chaque type de changement, est généré de manière flexible en fonction des stratégies d'évolution choisisses par les ontologistes. De cette manière, le modèle du CHB est capable de traiter des situations ouvertes et de prendre en compte toute une palette de stratégies d'évolution possibles.

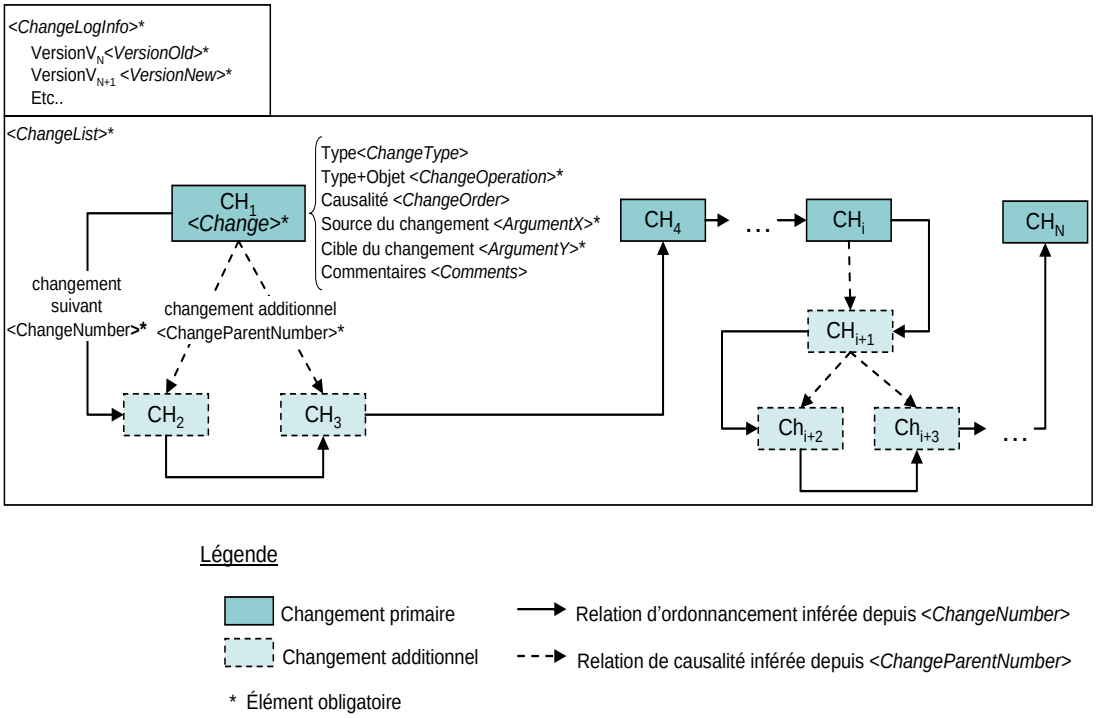


Figure IV-8. Le journal de type CHB présenté sous la forme d'un arbre organisé de changements

Finalement, le CHB peut être également considéré comme un document de spécifications pour étendre les fonctionnalités d'édition, offertes actuellement par les éditeurs d'ontologies. Par le fait qu'il offre une classification de changements – en fonction de leur type et de leur objet d'application – ainsi qu'une description de données d'entrées/sortie pour chaque changement (les `<ArgumentX>` et `<ArgumentY>`), il spécifie explicitement ce que les éditeurs d'ontologies doivent implémenter pour permettre l'édition des changements complexes, en plus des changements élémentaires.

4.2.1.3. **Formaliser les changements à l'aide du langage OC (étape 2)**

Le modèle du CHB peut être utilisé, par divers éditeurs OWL, pour journaliser, d'une manière normalisée, les changements élémentaires et complexes et obtenir ainsi une trace de changements décrits à l'aide des métadonnées du modèle. Néanmoins, si on attend des agents logiciels qu'ils soient capables d'interpréter et de raisonner sur cette trace alors on a besoin d'un langage qui puisse décrire formellement la signification des métadonnées, spécialement

les éléments <ArgumentX> et <ArgumentY> qui prennent un sens différent en fonction de l'opération de changement. Le langage *OntologyChange (OC)*, que nous proposons dans cette section, répond à ce besoin et procure un certain nombre de constructeurs langagiers pour la représentation formelle des métadonnées et des significations associées aux <ArgumentX> et <ArgumentY>. Nous commençons par montrer le rôle joué par le CHB dans la formalisation des changements. Nous présentons ensuite les principes à la base du langage OC. Finalement, nous décrivons les constructeurs langagiers OC et nous illustrons leur utilisation par quelques exemples.

4.2.1.3.1. *Rôle du CHB dans la formalisation des changements*

Lors de l'étape précédente, les changements sont récupérés durant l'évolution de l'ontologie en utilisant le modèle de journalisation du CHB. Dans cette étape, le système CHB formalise les données ainsi obtenues à l'aide du langage OC de représentation des changements (*cf.* Figure IV-5). Il produit alors une trace de changements représentés dans un langage ayant une sémantique formelle riche, capable de spécifier la signification de tout changement applicable aux ontologies OWL.

4.2.1.3.2. *Langage OntologyChange (OC) de représentation de changements*

Principes du langage OC

Les principes à la base du langage *OntologyChange* (OC) sont au nombre de trois : expressivité suffisante, vocabulaire minimal, langage orienté OWL.

Expressivité suffisante. Le langage devrait être d'une expressivité suffisante pour représenter une bonne partie de changements applicables aux ontologies OWL, que ce soit des changements élémentaires ou complexes. Il devrait aussi être capable de représenter formellement les stratégies d'évolution utilisées pour résoudre les inconsistances introduites par les changements, c'est-à-dire de rendre compte des relations entre un changement primaire et ses changements additionnels.

Vocabulaire minimal. Le vocabulaire du langage OC devrait être le plus minimal possible, sans réduire cependant ni son expressivité, ni sa capacité de représenter

intégralement la sémantique des opérations de changement. Ce principe trouve son explication dans le fait que le langage OC devrait être utilisé, voire comprit, par divers agents, logiciels et humains, qui s'engagent à utiliser les mêmes termes afin de s'échanger des informations sur les changements accomplis. Ainsi, plus ce vocabulaire est restreint, plus facilement il sera accepté et partagé par les agents.

Langage orienté OWL. La représentation des changements devrait se réaliser dans un langage fondé sur OWL afin de pouvoir profiter de la richesse sémantique de constructeurs de classes et de propriétés. En vérité, pourquoi introduire de nouveaux constructeurs si le langage OWL en possède déjà ? Comme nous le montrons dans le Tableau IV-7 et dans l'Appendice D, avec un nombre minimal de constructeurs, mais en utilisant ceux d'OWL, on peut représenter la palette entière des changements d'ontologie. Ce qui est particulièrement intéressant si on pense que, en enrichissant le langage OWL avec un petit nombre de constructeurs, il est possible de lui conférer des moyens pour exprimer, en plus de la conceptualisation d'un domaine, l'évolution de cette conceptualisation. Ceci rejoint d'ailleurs un des aspects soulignés par le consortium W3C : « un langage d'ontologie (OWL) doit pouvoir s'accommoder de la révision d'une ontologie » (W3C_WebOnt, 2004b).

Constructeurs du langage OC

Dans le Tableau IV-7, nous présentons les constructeurs du langage. Nous montrons ensuite comment ils peuvent être combinés aux constructeurs provenant du langage OWL pour exprimer formellement les changements apportés lors de l'évolution d'une ontologie.

Tableau IV-7 Constructeurs du langage OntologyChange (OC) et leur organisation pour représenter les changements

Constructeur du langage OC	Organisation de constructeurs OC
oc:ChangeTrace - élément pour déclarer et baliser une trace des changements.	**<oc:ChangeTrace>** <!--déclaration des changements. Par `ChangeOperation` nous désignons tout changement ayant un constructeur qui lui est spécifique --> **<oc:ChangeOperation>** **<oc:from>** <!--description de la source du changement (c.-à-d. classes, propriétés et/ou axiomes appartenant à V_N) en utilisant des constructeurs OWL--> **</oc:from>** **<oc:to>** <!--description de la cible du changement (c.-à-d. des classes, propriétés et/ou axiomes appartenant à V_{N+1}) en utilisant des constructeurs OWL--> **</oc:to>** <!-- définition des changements additionnels causés par le changement ci-dessus --> **<oc:additionalChanges>** <!-- Chaque changement additionnel est défini selon le même procédé : (1) déclaration du changement au moyen de constructeurs OC dédiés; (2) description du changement : source, cible ; (3) déclaration des changements additionnels, s'ils existent --> **</oc:additionalChanges>** **</oc:ChangeOperation>** **<oc:ChangeOperation/>** Etc. **<oc:ChangeOperation/>** <!-- La définition de la trace des changements s'achève par la balise suivante --> **</oc:ChangeTrace>**
oc:AddClass/Property; **oc:DeleteClass/Property**; **oc:MergeClasses**; **oc:ModifyPropertyRange**; etc. - éléments pour déclarer des changements bien particuliers ; ils sont les identifiants de classes de dernier niveau de la hiérarchie `ChangeOperation`.	
oc:from - cet élément déclare qu'il existe une source du changement, plus précisément il réfère aux valeurs des `<ArgumentX>` pour un changement spécifique. Pour décrire cette source, le langage OC permet l'utilisation exclusive des constructeurs propres à l'OWL (*cf.* Appendice D).	
oc:to - cet élément déclare qu'il existe une cible du changement, plus précisément il réfère aux valeurs des `<ArgumentY>` pour un changement spécifique. Pour décrire cette source, le langage OC permet l'utilisation exclusive des constructeurs propres à OWL (*cf.* Appendice D)	
oc:additionalChanges - cet élément déclare qu'ils existent des changements additionnels requis pour résoudre les inconsistances introduites par un autre changement d'ontologie. Il s'utilise comme balise pour la description détaillée de l'ensemble de changements additionnels.	

Représentation de changements à l'aide d'OC+OWL

Le langage OC pour la représentation formelle de changements utilise un nombre minimal de constructeurs qui lui sont propres et un bon nombre de constructeurs OWL pour exprimer les classes et leurs axiomes, ainsi que les propriétés et leurs caractéristiques. Ainsi, les constructeurs OC sont utilisés par le CHB uniquement pour déclarer le type de changements et pour délimiter la description de leur source et de leur cible, cette description étant représentée strictement à l'aide des constructeurs OWL. Dans cette section, nous allons illustrer, à l'aide de quelques exemples, comment les constructeurs OWL sont utilisés par imbrication avec ceux du langage OC pour représenter formellement des changements ontologiques récupérés à l'aide du modèle du CHB (dans l'Appendice D, le lecteur pourra trouver un plus grand nombre d'exemples). Précisons que les termes '*old*' et '*new*', qui préfixent les noms de classes et de propriétés, sont des identifiants d'ontologies ; ils expriment respectivement l'URI de version V_N et V_{N+1} (*cf.* explication donnée au paragraphe 4.2.1.4.1.).

Exemple 1 : Représentation d'un changement qui ajoute une nouvelle classe en V_{N+1}

```
<oc:AddClass>
  <oc:to>
    <owl:Class rdf:about="etnew;#Doctorant">
      <rdfs:subClassOf rdf:resource="etnew;#PersonnelEtudiant"
/>
    </owl:Class>
  </oc:to>
  <rdfs:comment>Ajout Doctorant au PersonnelEtudiant</rdfs:comment>
</oc:AddClass>
```

Exemple 2 : Représentation d'un changement qui modifie le codomaine d'une propriété en V_{N+1}

```
<oc:ModifyPropertyRange>
  <oc:from>
    <owl:ObjectProperty rdf:about="etold;#demandeAideActiviteA">
      <rdfs:range rdf:resource="old;#Pedagogue"/>
    </owl:ObjectProperty>
  </oc:from>
  <oc:to>
    <owl:ObjectProperty rdf:about="etnew;#demandeAideActiviteA">
      <rdfs:range rdf:resource="etnew;#AnimateurActivites"/>
    </owl:ObjectProperty>
  </oc:to>
  <rdfs:comment>Modification du codomaine</rdfs:comment>
</oc:ModifyPropertyRange>
```

Exemple 3 : Représentation d'un changement qui : (a) fusionne deux classes dans une autre; (b) est accompagné de deux changements additionnels (modification d'une superclasse et déplacement d'un axiome).

Changement primaire

```
<oc:MergeClasses>
  <oc:from rdf:parseType="Colection">
    <owl:Class rdf:about="etold;#DesignerPedagogique"/>
    <owl:Class rdf:about="etold;#Professeur"/>
  </oc:from>
  <oc:to>
    <owl:Class rdf:about="etnew;#ConcepteurCours">
      <rdfs:subClassOf rdf:resource="etnew;#P"/>
    </owl:Class>
  </oc:to>
```

Changements additionnels

```
<oc:additionalChanges>

<oc:ModifySuperClass>
  <oc:from>
    <owl:Class rdf:about="etold;#Designer_MISA">
      <rdfs:subClassOf rdf:resource="etold;#DesignerPedagogique"/>
    </owl:Class>
  </oc:from>
  <oc:to>
    <owl:Class rdf:about="etnew;#Designer_MISA">
      <rdfs:subClassOf rdf:resource="etnew;#ConcepteurCours"/>
    </owl:Class>
  </oc:to>
</oc:ModifySuperClass>

<oc:MoveDisjointClass>
  <oc:from>
    <owl:Class rdf:about="etold;#Professeur">
      <owl:disjointWith rdf:resource="etold;#Pedagogue"/>
    </owl:Class>
  </oc:from>
  <oc:to>
    <owl:Class rdf:about="etnew;#ConcepteurCours">
      <owl:disjointWith rdf:resource="etnew;#Pedagogue"/>
    </owl:Class>
  </oc:to>
</oc:MoveDisjointClass>

</oc:additionalChanges>

<rdfs:comment>Fusion des classes et changements additionnels
</rdfs:comment>

</oc:MergeClasses>
```

4.2.1.4. **Archiver les changements dans la V_{N+1}Change (étape 3)**

Comme discuté auparavant, le fait de pouvoir identifier les changements à partir uniquement des versions d'ontologie, sans avoir besoin des fichiers-journal externes, constitue un avantage majeur dans le contexte du Web sémantique, tout comme celui d'avoir une vue complète sur l'évolution et sans aucune ambiguïté sémantique. La solution que nous

proposons pour répondre à ces deux aspects est celle d'introduire, dans la nouvelle version de l'ontologie, la trace des changements formalisés à l'aide du langage combiné OC+OWL, trace qui exprime l'évolution de V_N à V_{N+1}. Nous nommons **V_{N+1}Change** cette nouvelle version avec trace des changements. Ainsi, quand on accède à la version V_{N+1}Change on accède à la conceptualisation spécifiée par l'ontologie, mais également à l'évolution de cette ontologie, à l'histoire de la conceptualisation du domaine.

4.2.1.4.1. *Rôle du CHB dans l'archivage des changements : obtention de la V_{N+1}Change*

Nous avons choisi d'intégrer la trace de changements d'une manière qui préserve la version V_{N+1}Change compatible avec OWL. Pour cela, le système CHB introduit la trace des changements formalisés dans la version V_{N+1}Change en utilisant le constructeur prédéfini `owl:versionInfo`. Ce constructeur, dont l'objet est généralement une chaîne fournissant des renseignements à propos des versions d'ontologie, ne contribue pas à la signification logique de l'ontologie et n'a donc aucun impact autre qu'informatif. Nous obtenons ainsi la version V_{N+1}Change qui conserve sa consistance à l'égard d'OWL, tout en offrant la possibilité d'identifier les changements élémentaires et complexes, ainsi que les stratégies d'évolution appliquées, sans recourir à aucune autre source d'information externe.

Nous donnons un exemple d'une version de type V_{N+1}Change dans la Figure IV-10. Ici, les préfixes `old` et `new` sont introduits par le CHB à l'aide de deux déclarations d'entités (W3C_Recommendation, 2006), le but étant de faciliter l'écriture des éléments appartenant à la version V_N (`…OntologieActeur/v2`) et à la V_{N+1} (`…/OntologieActeur/v3`)[40] (*cf.*Figure IV-9).

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
     <!ENTITY old "http://example.org/OntologieActeur/v2" >
     <!ENTITY new "http://example.org/OntologieActeur/v3" >
     <!ENTITY xsd  "http://www.w3.org/2001/XMLSchema#" >
   ]>
```

Figure IV-9. Déclaration des préfixes `old` et `new`

[40] `etold;#Professeur` se développerait en `http://…OntologieActeur/v2#Professeur`, les deux expressions ayant cependant la même signification.

Figure IV-10. Exemple de la version V_{N+1}Change produite par le CHB (rappelons que la trace intégrée des changements est délimitée par les balises `<oc:ChangeTrace>`)

La solution de représenter les changements à l'intérieur de chaque nouvelle version d'une ontologie nous dispense d'utiliser des mécanismes de localisation et d'accès aux fichiers-journal, autres que ceux utilisés pour les versions d'ontologie. De plus, le fait que toute version V_{N+1}Change est une version compatible OWL, nous permet d'utiliser les fonctionnalités d'emmagasinage et de navigation fournies par les librairies d'ontologies distribuées. Étant donné que la trace des changements est intégrée à la version V_{N+1}Change, tout en étant (semi)compatible avec OWL, nous permet d'adapter les outils de visualisation d'ontologie pour permettre également la visualisation des changements apportés aux ontologies. Dans cette thèse, nous n'avons ni utilisé des librairies d'ontologies pour l'archivage des versions de type V_{N+1}Change, ni adapté des outils existants pour permettre la visualisation de changements. Nous pensons cependant que ces deux aspects sont tout à fait envisageables comme perspectives de travail.

4.2.2. **Identification des changements à partir de l'analyse de V_{N+1}Change**

Le système CHB possède deux fonctionnalités de base. La première consiste dans la journalisation de changements en trois étapes : la capture, la formalisation et l'intégration des changements dans la version V_{N+1}Change. La deuxième fonctionnalité, présentée dans cette section, consiste dans l'identification des changements apportés à une version d'ontologie V_N pour obtenir une nouvelle version V_{N+1}. Pour cela, le système CHB récupère la version V_{N+1}Change, interprète la trace intégrée des changements et l'affiche dans une interface de visualisation des changements (*cf.* Figure IV–11). Cette interface présente les changements élémentaires et complexes sous une forme hiérarchique, en fonction de la relation de causalité (primaire/additionnel) qui existe entre eux. Les identifiants de la version V_N et V_{N+1} sont affichés en haut de la fenêtre de visualisation ; ils précisent le chemin d'accès aux versions, au cas où l'utilisateur voudrait les consulter.

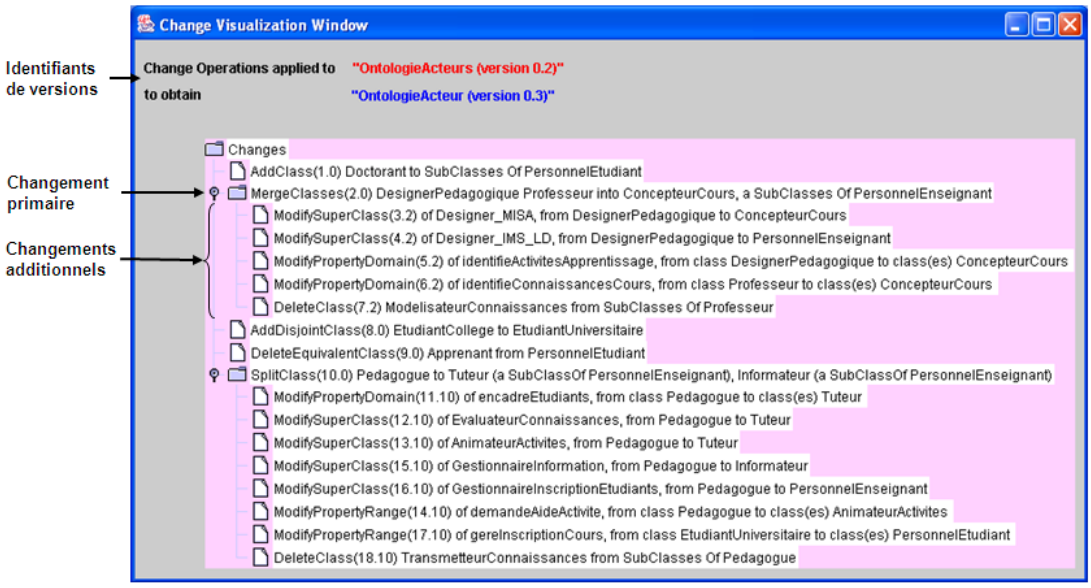


Figure IV-11. Interface du CHB pour la visualisation des changements

Étant donné que les changements sont représentés d'une manière riche sémantiquement à l'aide du langage OC, l'aspect le plus important de cette fonctionnalité ne réside pas dans la récupération des changements, mais plus dans la manière de les présenter à l'interface afin de faciliter la compréhension des utilisateurs. L'interface actuelle du système CHB possède de faibles critères ergonomiques, mais comme le but de cette thèse est de valider de nouvelles idées, implémentées dans le prototype exploratoire CHB, cette interface de visualisation convient momentanément. Cependant, une des perspectives intéressantes de la thèse, une fois validée l'utilité du CHB, serait de développer un outil de visualisation plus riche, permettant la présentation des changements sous diverses formes.

4.3 **Conclusion**

Dans ce chapitre, nous avons répondu au deuxième objectif de cette thèse. Nous avons proposé le système *ChangeHistoryBuilder* (CHB) pour tracer l'histoire des changements apportés aux versions d'une ontologie, système qui constitue le premier module de notre cadre de référence pour la gestion des changements.

Dans un premier temps, nous avons présenté le modèle de journalisation du CHB fondé, en partie, sur l'ontologie des changements. Ce modèle, composé d'un ensemble des métadonnées, permet aux différents éditeurs de journaliser, d'une manière uniformisée, cohérente et complète, des changements élémentaires et complexes, de même que leurs relations de causalité (primaire/additionnel).

Dans un deuxième temps, nous avons proposé le langage *OntologyChange* (OC) pour une formalisation standardisée des changements. Ce langage est fondé sur un nombre (minimal) de constructeurs `oc`, mais utilise aussi les constructeurs `owl` pour exprimer des classes, des axiomes et des propriétés d'une manière riche et consistante. Le but est de permettre à tout agent informatique, capable de manipuler le formalisme OC+OWL, d'interpréter la trace des changements journalisée à l'aide du modèle du CHB. C'est ce que fait d'ailleurs le système SAM, que nous présentons dans le chapitre suivant (Chapitre V).

Dans un troisième temps, nous avons décrit le moyen utilisé par le système CHB pour éviter les problèmes d'accès aux fichiers-journal distribués. En effet, il insère chaque trace des changements à l'intérieur de la version V_{N+1}Change, au moyen de la propriété d'annotation `owl:VersionInfo`. Ceci préserve la version conforme à OWL, mais offre, en même temps, la possibilité d'avoir accès à l'histoire des changements effectués pour obtenir cette version.

Finalement, nous avons montré comment, à tout moment après l'évolution, le CHB interprète la V_{N+1}Change et affiche les changements à l'interface du système.

Dans le prochain chapitre, nous parlons du deuxième module du cadre de référence pour la gestion des changements. Il s'agit du système SAM qui, en interprétant la V_{N+1}Change et en extrayant les changements, est capable de fournir des recommandations pertinentes à la modification du référencement sémantique, affecté par l'évolution d'une ontologie.

Chapitre V

REFERENCEMENT SÉMANTIQUE ÉVOLUTIF FONDÉ SUR L'ANALYSE DES CHANGEMENTS APPORTÉS AUX VERSIONS D'ONTOLOGIE

Nous avons présenté dans les chapitres précédents une partie du cadre de référence que nous proposons pour la gestion des changements apportés aux versions d'ontologie. Ce cadre se compose de deux modules principaux : le *ChangeHistoryBuilder* (CHB) avec son ontologie des changements et le *SemnaticAnnotationModifier* (SAM) que nous présentons dans ce chapitre.

Le système SAM se veut une réponse à notre troisième objectif de recherche qui vise à identifier les effets des changements sur le référencement sémantique des ressources et à développer des solutions pour résoudre les effets problématiques, menant jusqu'à une perte d'accès aux ressources par exemple.

Avant de présenter le fonctionnement du SAM, nous mettons en évidence le manque d'études approfondies sur la maintenance du référencement fondé sur des ontologies évolutives (*cf.* Section 5.1.1.). Nous discutons ensuite (*cf.* Section 5.2) de la première fonctionnalité de SAM, nommée **Analyse des Changements**, dont le but est d'analyser l'impact de l'évolution sur l'accès et l'interprétation des ressources référencées par des classes provenant des ontologies évolutives. Nous présentons également la deuxième fonctionnalité du SAM, nommée **Modification du Référencement Sémantique** (*cf.* Section 5.3). Cette fonctionnalité permet la modification du référencement sémantique pour le

rendre compatible avec la nouvelle version de l'ontologie, mais aussi pour résoudre les problèmes d'accès et d'interprétation des ressources référencées. Nous terminons ce chapitre en résumant nos apports (*cf.* Section 5.4).

Les travaux présentés dans ce chapitre ont fait l'objet d'une publication (Rogozan et Paquette, 2005b) et d'une communication (Rogozan, 2004b).

5.1 **Changements ontologiques et référencement sémantique évolutif**

L'évolution fait partie intégrante du cycle de vie des ontologies. Cette évolution se traduit par l'application des changements à une version d'ontologie (V_N) pour en obtenir une nouvelle version (V_{N+1}), tout en veillant à maintenir la consistance[41] de cette nouvelle version, mais aussi l'intégrité de ce qui repose déjà sur l'ontologie. En effet, lorsque l'ontologie change, ses changements ont un impact sur tout ce qui a été construit au-dessus, en particulier sur le référencement sémantique des ressources. Dans cette section, nous discutons d'abord des recherches actuelles portant sur l'étude des effets des changements, leurs apports et leurs faiblesses respectives. Ensuite, nous mettons en évidence le manque total d'approches pour maintenir l'accès et l'interprétation des ressources référencées après l'application d'un certain nombre de changements problématiques. Nous concluons cette section par une présentation brève du système *SemanticAnnotationModifier*, le deuxième module du cadre de référence que nous proposons pour la gestion des changements apportés aux versions d'ontologie.

5.1.1. **Étude de l'existant**

5.1.1.1. **Analyse des effets des changements sur le référencement sémantique**

À notre connaissance, il n'existe actuellement ni études théoriques complètes, ni outils capables d'analyser les effets des changements sur le référencement sémantique des ressources. Seuls les premiers auteurs cités ci-après proposent une vue sur les effets de certains changements élémentaires. Ils restent néanmoins très brefs dans leur analyse.

Ainsi, Heflin et Hendler, (2000), analysent les effets des changements qui ajoutent ou qui effacent des classes et des propriétés. Ils mettent en évidence le fait que tout changement d'effacement peut conduire à une altération de l'accès aux ressources référencées, et que tout changement d'ajout n'affecte ni l'interprétation, ni l'accès aux ressources référencées. Cette

[41] Une ontologie est consistante si les contraintes concernant le modèle et ses axiomes sont respectées.

analyse est toutefois assez incomplète, étant donné que les auteurs abordent uniquement les changements élémentaires et qu'ils ne distinguent pas le changement de classe de celui de propriété, sachant que l'effet produit par l'un ou par l'autre peut être différent.

Oliver, Shahar, Musen et Shortliffe, (1999), soulignent les changements qui affectent ou non la hiérarchie des classes, tout comme Stuckenschmidt et Klein, (2003b), qui indiquent, en plus, les situations où les classes deviennent plus spécialisées ou plus généralisées sous l'effet des changements qui ajoutent ou qui effacent des entités d'une ontologie. Toutefois, ces auteurs traitent uniquement les effets des changements élémentaires, ce qui est insuffisant pour une interprétation pertinente de la modification conceptuelle introduite dans la nouvelle version de l'ontologie.

Finalement Sunagawa, Mizoguchi et *al*., (2003), classifient les changements qui ajoutent ou qui effacent des classes ou des propriétés (attributs) en fonction de mesures à prendre pour préserver ou non la dépendance entre l'ontologie évoluée et d'autres ontologies qui en dépendent. Du même point de vue, admettent l'existence des effets sur la signification des ontologies dépendantes. Ces auteurs proposent de préserver la consistance de ces dernières par une propagation des changements de l'ontologie évoluée vers les ontologies dépendantes. Précisons cependant que Maedche, Motik et Stojanovic, (2003), ne traitent pas les effets des changements sur le référencement sémantique des ressources. Ils n'offrent donc aucune solution ni pour leur analyse, ni pour la résolution des effets plus problématiques.

5.1.1.2. **Mise à jour du référencement sémantique**

Une gamme assez large d'outils de référencement sémantique des ressources est proposée à l'heure actuelle.

Par exemple, les outils COHESE (Bechhofer et Goble, 2001) et MnM (Motta, Vargas-Vera, Domingue, Lanzoni, et Ciravegna, 2002), qui facilitent le référencement des pages web en utilisant des balises HTML construites à partir des ontologies. Ou encore, l'outil SHOE (Heflin, 2001) qui permet aux utilisateurs de décrire manuellement diverses pages web avec des instances d'ontologies décrites avec SHOE, un langage qui ajoute au HTML des marqueurs spécifiques. L'outil l'OOF (Collier et al., 2004), offre un environnement intégré

pour la création des ontologies et des références sémantiques fondées sur ces ontologies. Le projet du consortium W3C, Annotea (Koivunen, 2005), est une architecture RDF permettant le référencement partagé des ressources du Web, comme les documents HTML ou XML, en suivant un protocole[42] spécifique de travail. Finalement, le modèle d'architecture CREAM (Handschuh, Staab, et Maedche, 2001) souscrit aussi aux formats standards du W3C avec des références représentées en RDF (ou OWL) pour décrire des documents HTML/XML. Il spécifie également les composants exigés pour supporter la création semi-automatique des références fondées sur une ontologie.

Parmi ces outils, aucun ne possède cependant des fonctionnalités pour la gestion du référencement sémantique, ni pour la maintenance des liens de référence entre les ressources et des ontologies évolutives. Seul le modèle d'architecture CREAM propose des recommandations générales à cet effet, mais sans donner de piste de solution (Handschuh, 2007).

5.1.2. **Référencement sémantique évolutif avec SAM**

Dans ce chapitre, nous abordons notre troisième objectif de recherche, à savoir : développer une méthode et un outil – **SemanticAnnotationModifier** (**SAM)** – pour (1) analyser les effets des changements sur le référencement sémantique de ressources et (2) maintenir l'accès et l'interprétation des ressources référencées à la suite de l'évolution des ontologies utilisées comme référentiels sémantiques. Le système SAM est le deuxième module du notre cadre de référence pour la gestion des changements apportés aux versions d'ontologie.

Dans un premier temps, nous mettons en évidence ce que nous comprenons par le référencement sémantique d'une ressource. Dans un second temps, nous présentons brièvement les deux fonctionnalités de base du système SAM.

---

[42] Annotea fournit, à l'aide de XPointer, une méthode permettant de localiser des annotations dans un document. XPointer est une recommandation du W3C pour identifier des fragments de ressources avec leur URI.

5.1.2.1. **Référencement sémantique au moyen des URI/UKI**

La Figure V-1 illustre ce que nous comprenons par la notion de référencement sémantique. Ainsi, le **référencement sémantique** d'une ressource se compose d'une ou de plusieurs références sémantiques, associées à la ressource afin de décrire son contenu ou son utilisation. Une **référence sémantique** est précisément l'identifiant d'une connaissance formelle, exprimée sous la forme d'une classe dans une ontologie. Chaque référence sémantique est spécifiée formellement par ce qu'on appelle un **UKI** (*Uniform Knowledge Identifier*).

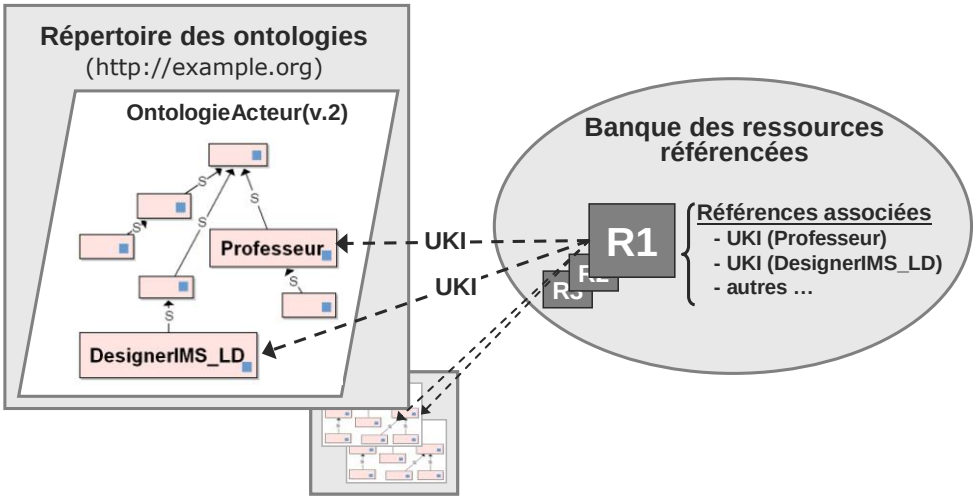


Figure V-1. Référencement sémantique d'une ressource

Pour expliciter les références sémantiques, nous utilisons la syntaxe générale des URI (*Uniform Ressource Identifier*) (Berners-Lee, 2005). Comme exemplifié dans la Figure V-2, un URI est une séquence compacte des caractères qui identifient une ressource à l'aide d'un ensemble des composants : (1) le `schème`, qui détaille comment les identifiants de ce schéma sont alloués ; (2) l'`autorité`, qui gouverne l'interprétation du reste de l'URI ; (3) le `chemin`, qui identifie une ressource au sein du domaine d'application du schéma et de l'autorité de nommage ; (4) le `fragment`, qui identifie, au besoin, une partie spécifique de la ressource identifiée par les trois composants précédents.

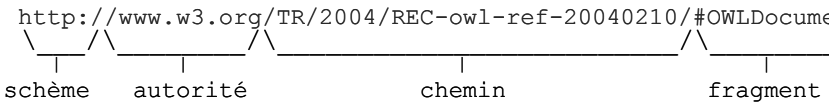


Figure V-2. Exemple d'URI et ses quatre composants

Selon la compréhension sous-jacente aux URI, toute chose ayant une identité peut être considérée comme une ressource, que ce soit une ressource physique (image, document, service, acteur) ou abstraite (concepts dans une ontologie). Cependant, dans notre contexte nous voulons affirmer que les références sémantiques identifient uniquement des classes provenant d'une ou plusieurs ontologies. Pour cela, nous introduisons le terme **UKI (*Uniform Knowledge Identifier*)** et nous le définissons comme étant un URI avec la particularité que les trois premiers composants doivent nécessairement identifier une ontologie (ou une version d'ontologie) tandis que le quatrième doit identifier une classe à l'intérieur de cette ontologie (*cf.* Figure V-3). Un UKI se compose donc de deux parties principales :

- l'identifiant d'une ontologie (ou version d'ontologie) composé d'un `schème`, d'une `autorité` et d'un `chemin`. Cet identifiant est spécifiquement l'**URI de l'ontologie** (ou version d'ontologie). Bien qu'il puisse être intéressant de déduire la nature de l'ontologie ainsi identifiée en analysant les composants de l'URI, nous n'allons pas le faire dans cette thèse. Au contraire, nous allons considérer l'URI de l'ontologie (ou version d'ontologie) comme une chaîne compacte et opaque à tout traitement de décomposition[43] d'URI.

- le **nom d'une classe** appartenant à l'ontologie précédemment identifiée, représenté au moyen d'un identifiant de `fragment`. De cette manière, l'UKI (URI d'ontologie + nom de classe) permet l'identification exacte d'une classe en se référant à une ontologie et à une information d'identification additionnelle qui spécifie un élément à l'intérieur de cette ontologie.

[43] Dans la littérature de spécialité, ce traitement est dénommé aussi déréférencement d'URI.

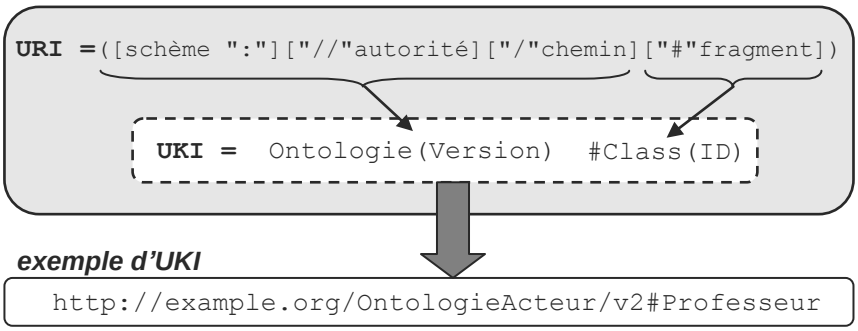


Figure V-3. Un UKI est un URI de l'ontologie/version d'ontologie (p.ex. `http://example.org/OntologieActeur/v2`) **avec un identifiant de fragment qui est le nom d'une classe** (p.ex. `Professeur`)**.**

5.1.2.2. **Fonctionnalités de SAM : analyse des changements et modification du référencement**

Sachant que la préservation de l'accès et de l'interprétation des ressources référencées par des ontologies est un aspect essentiel du Web sémantique, le manque de méthodes et d'outils pour la supporter devient un problème majeur à l'heure actuelle. En conséquence, nous proposons **SemanticAnnotationModifier (SAM)** comme une solution pour assurer un référencement sémantique évolutif des ressources décrites à l'aide des concepts (classes) d'une ontologie qui évolue dans le temps. La première idée à la base du système SAM est de permettre la modification du référencement sémantique des ressources d'une manière qui soit représentative de l'évolution, c'est-à-dire en fonction des changements apportés à la version V_N pour obtenir V_{N+1}. La deuxième idée est celle de considérer SAM comme un système conseiller, qui guide les utilisateurs lors du processus de modification du référencement, mais qui ne prend aucune décision automatique.

Précisons que, dans le contexte du SAM, le terme 'utilisateur' désigne une palette d'acteurs partageant le même rôle : celui de gestionnaire de ressources référencées par des concepts provenant des ontologies évolutives. Parmi ces acteurs, nous retrouvons, par exemple, des concepteurs d'ontologie, des responsables de banques de ressources, des gestionnaires de systèmes dont les composants sont référencés.

Le système SAM possède deux fonctionnalités principales, comme nous l'illustrons dans la Figure V–4.La **première fonctionnalité** fournit aux utilisateurs une **analyse des effets des changements** sur le référencement sémantique de ressources. Le but ici est de mettre en évidence les effets problématiques allant jusqu'à une perte totale d'accès aux ressources référencées ou jusqu'à une interprétation inconsistante de la description attachée aux ressources au moyen de concepts provenant de l'ontologie. Nous présentons cette fonctionnalité en détail dans la Section 5.2.

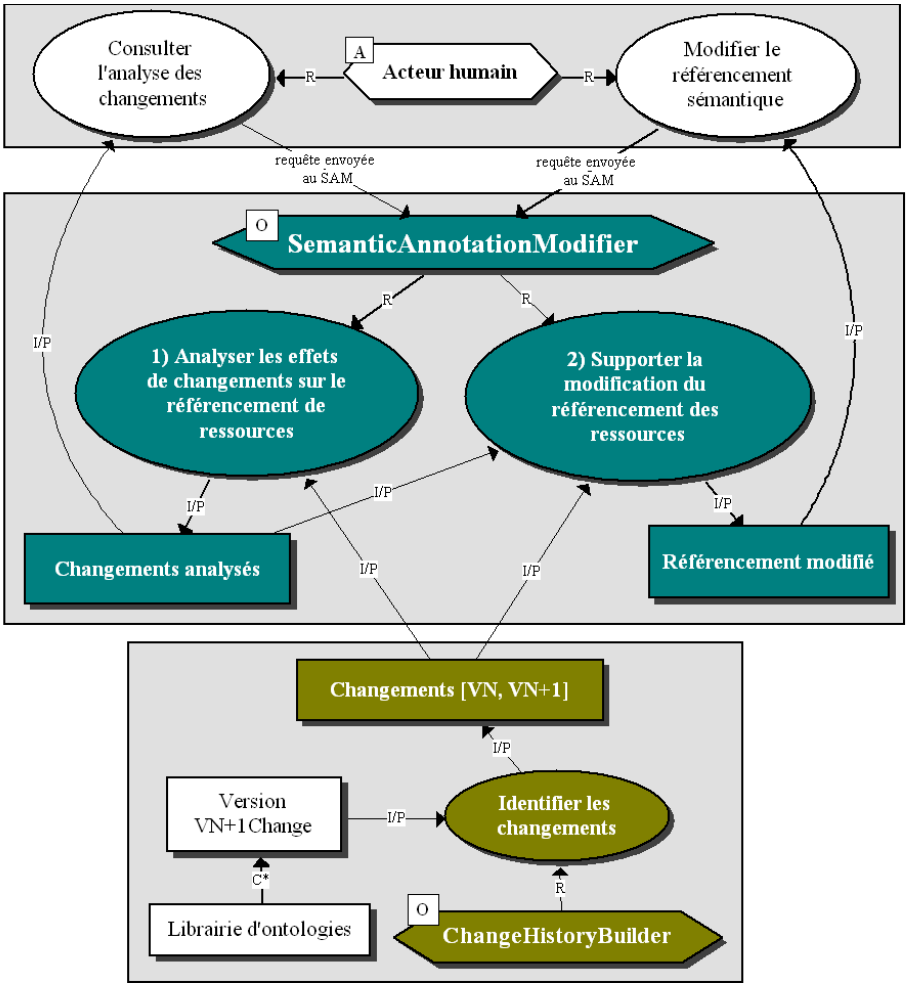


Figure V-4. Les deux fonctionnalités du SAM
(La légende du formalisme graphique est consultable dans l'Appendice A)

La **deuxième fonctionnalité** du système SAM fournit aux utilisateurs un support à la **modification du référencement sémantique** des ressources, affecté par les changements. Nous la présentons en détail dans la Section 5.3. Le but principal de cette fonctionnalité est de préserver l'accès aux ressources via V_{N+1}. Soit, par exemple, une ressource R référencée uniquement par une classe C. Si on efface cette classe en V_{N+1}, alors la ressource R n'est plus accessible au moyen des requêtes adressées à l'ontologie évoluée. Pour préserver l'accès à cette ressource, il faudrait remplacer la classe C par une autre classe appartenant à V_{N+1}. Le but secondaire de cette deuxième fonctionnalité est celui de préserver la cohérence logique du référencement (p.ex. la déclaration d'une disjonction entre deux classes rend incohérente la description d'une ressource référencée à la fois par ces deux classes).

Comme pour le CHB, le système SAM a été développé selon une approche de programmation orientée objet avec le langage JAVA. Comme il s'agit toujours d'un prototype exploratoire, nous nous sommes, ici aussi, concentrée sur la conceptualisation sous-jacente au système plus que sur son implémentation. De ce fait, nous n'avons élaboré ni des programmes complexes, ni des interfaces conviviales, mais plutôt des solutions dont nous avons fait la preuve qu'elles sont soutenables informatiquement. C'est pourquoi nous ne donnons pas de détails d'implémentation, autres que les interactions entre le système et les utilisateurs.

5.2 **Analyser les changements avec SAM**

Dans l'analyse des changements, le principal rôle du SAM est d'interpréter le référencement sémantique et d'identifier les effets des changements sur l'accès aux ressources référencées (les ressources restent accessibles via V_{N+1}) et sur leur interprétation (les ressources sont correctement interprétables via V_{N+1}).

Tel qu'illustré dans la Figure V-5, SAM effectue une analyse des changements à la demande des utilisateurs qui désirent connaître les effets des changements qui affectent le référencement sémantique des ressources.

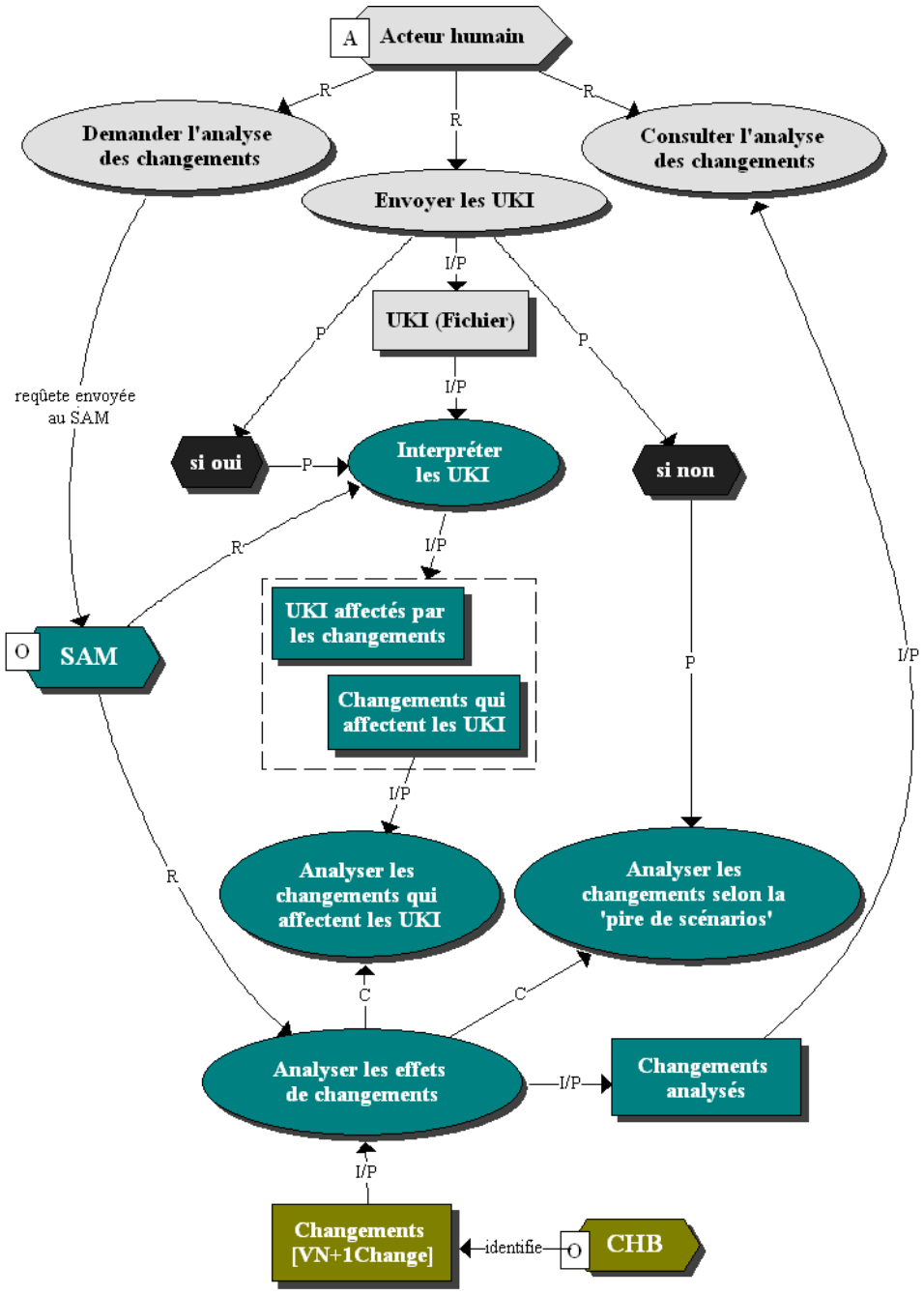


Figure V-5. Analyser les changements avec SAM
(La légende du formalisme graphique est consultable dans l'Appendice A)

Tout utilisateur a deux options possibles dans l'analyse des changements. La première lui permet de consulter une analyse générale, sans qu'il soit obligé d'envoyer un fichier des

UKI. Dans ce cas, le système SAM fournit une analyse où le *pire des scénarios*[44] est présenté pour chaque changement, c'est-à-dire qu'on considère toujours le pire des effets que les changements pourraient avoir sur le référencement sémantique des ressources. L'utilisateur doit préciser toutefois la version V_N de l'ontologie afin que SAM puisse récupérer les changements apportés à V_N pour obtenir V_{N+1} (voir la Section 5.2.2.2.

La deuxième option permet à l'utilisateur de consulter une analyse contextuelle, en fonction du référencement spécifique d'une ou de plusieurs ressources. Dans ce cas, seuls les changements qui affectent ce référencement sont mis en évidence. Les autres changements, bien que problématiques dans d'autres contextes, n'ont aucun effet dans celui-ci (p.ex. l'effacement d'une classe qui n'est pas utilisée dans le référencement qui nous intéresse). Pour cette deuxième option, l'utilisateur doit envoyer au SAM un fichier des UKI spécifiant les références sémantiques reliées aux diverses ressources. SAM interprète ensuite les UKI, identifie les changements[45] qui les affectent et analyse ensuite uniquement ces changements.

Dans les sections suivantes, nous présentons en détail la fonctionnalité d'analyse des changements du système SAM. Nous commençons par montrer les choix que nous avons faits pour que SAM puisse avoir accès au fichier des UKI ainsi que la forme d'organisation de ce fichier (*cf.* Section 5.2.1.). Ensuite, nous discutons comment SAM interprète et met en correspondance les UKI avec les changements apportés aux versions d'ontologie (*cf.* Section 5.2.2.). Finalement, nous illustrons l'analyse des changements fournie par SAM, en soulignant principalement les effets des changements sur le référencement sémantique des ressources (*cf.* Section 5.2.3.).

5.2.1. **Envoyer/Accéder aux UKI**

Lors de l'analyse des changements, les utilisateurs rendent disponible le référencement sémantique des ressources sous la forme d'un fichier contenant des UKI. Vu que le Web sémantique est un environnement essentiellement distribué, où chaque organisation d'agents

[44] De l'anglais '*worst-case scenario*', l'expression le pire des scénarios a ici la même signification.
[45] Pour l'identification des changements, le système SAM envoie une requête au système CHB avec l'identifiant de la version V_N. Il lui demande ensuite de lui fournir la V_{N+1}Change qui comporte la trace intégrée des changements apportés à V_N pour obtenir V_{N+1}.

assure son propre contrôle sur les ontologies et les ressources qui y réfèrent, il est impossible pour un système de parcourir toutes les banques de ressources afin de récupérer le référencement par lui-même. Une solution bien plus appropriée, que nous avons d'ailleurs appliquée dans cette thèse, est celle permettant à chaque utilisateur de fournir au SAM les UKI associés aux ressources, et cela, dans un fichier joint à leur demande d'analyse des changements. SAM offre alors une fonction de sélection à partir du menu, permettant aux utilisateurs d'introduire le fichier, comme illustré dans la Figure V-6.

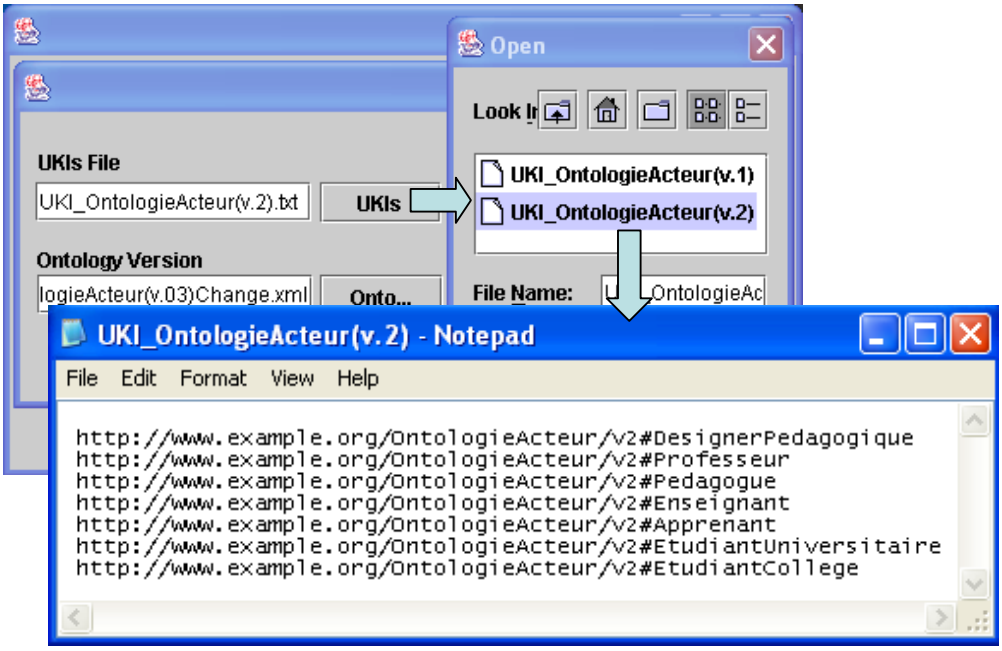


Figure V-6. Sélection et format d'un fichier des UKI

Une autre question concerne la forme qui prendra ce fichier. Pour ce prototype du système SAM, nous avons choisi d'éviter la diversité des formes possibles en introduisant un certain nombre de contraintes : (1) le fichier des UKI doit provenir d'un même propriétaire ; (2) il doit être organisé sous la forme d'une liste ; (3) tous les UKI doivent référer à une même version d'ontologie. Un exemple d'un tel fichier est fourni dans la Figure V-6.

5.2.2. **Interpréter les UKI**

Pour interpréter le fichier des UKI, SAM décompose chaque UKI afin de mettre en évidence l'URI de la version d'ontologie et le nom de classe utilisée comme référence.

Ensuite, il extrait les changements apportés à la version V_N, en demandant au CHB de lui fournir l'historique de l'évolution de V_N à V_{N+1}. Finalement, il met en correspondance les UKI avec les changements ainsi extraits. Le but ici est d'identifier quels sont les UKI affectés et quels sont les changements qui les affectent. La Figure V-7 illustre cette démarche.

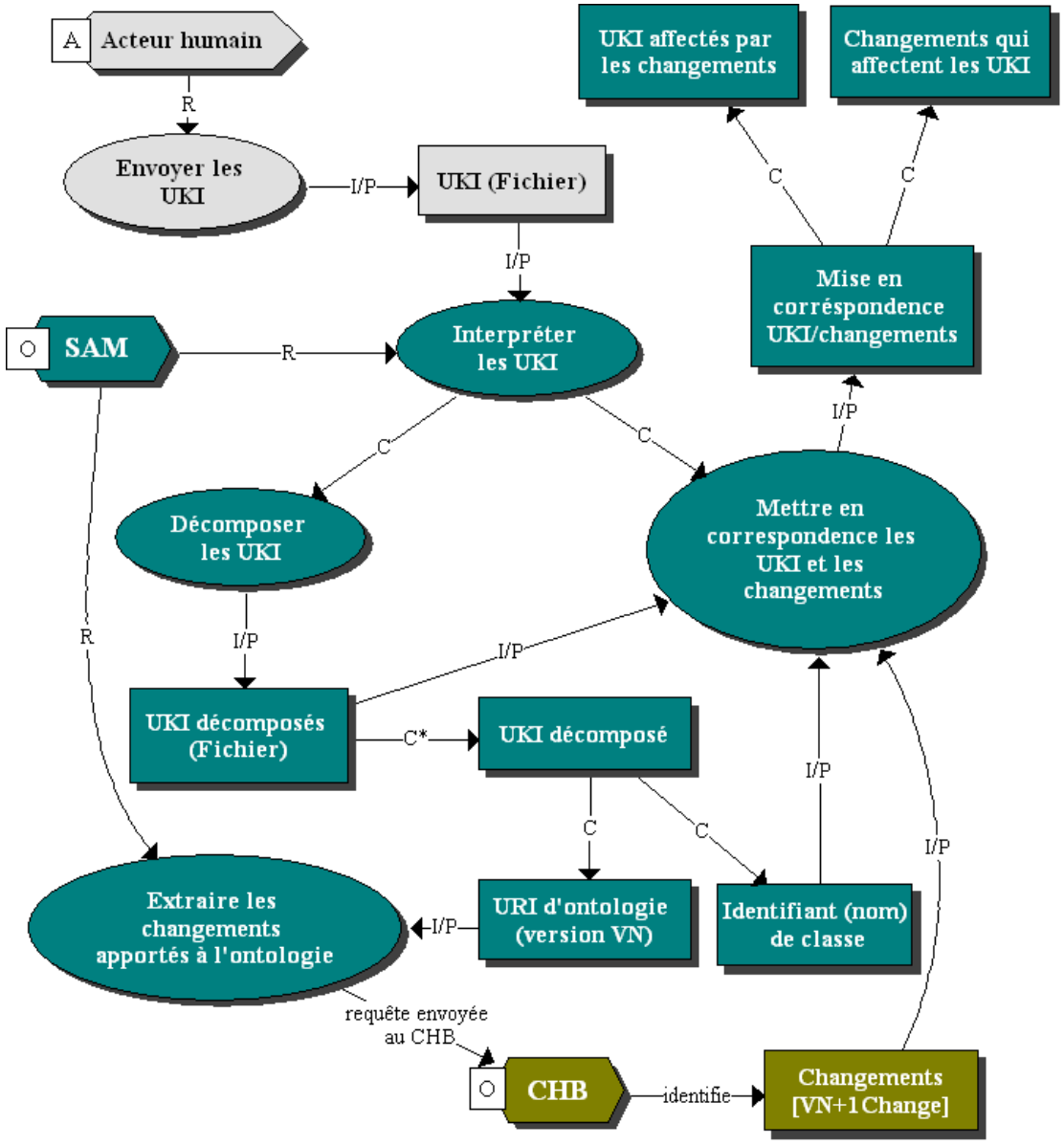


Figure V-7. Interpréter les UKI
(La légende du formalisme graphique est consultable dans l'Appendice A)

5.2.2.1. **Décomposer les UKI**

Nous avons déjà défini ce qu'est un UKI (*Uniform Knowledge Identifier*). Rappelons toutefois qu'un UKI est une chaîne compacte de caractères qui identifie, de manière claire et unique, une référence sémantique dans une ontologie.

Pour être capable d'extraire les changements et d'analyser leurs effets, SAM doit premièrement identifier la version de l'ontologie à l'intérieur d'un UKI (les composantes *schème*, *autorité* et *chemin*), mais aussi le nom de la classe qui sert de référence sémantique (l'identifiant de *fragment*). La décomposition d'un UKI dans toutes ses parties composantes est un processus complexe et problématique. Pour l'éviter, nous considérons l'URI de la version d'ontologie comme étant opaque au processus de décomposition. Ainsi, toute séquence des caractères avant le signe # est considérée comme étant l'URI de la version, et toute séquence après le signe # est considérée comme étant le nom d'une classe à l'intérieur de cette version identifiée. En suivant ce principe, le système SAM produit un fichier des UKI décomposés, tel qu'illustré dans la Figure V-8, où la version d'ontologie ainsi que le nom de classe sont bien différenciés.

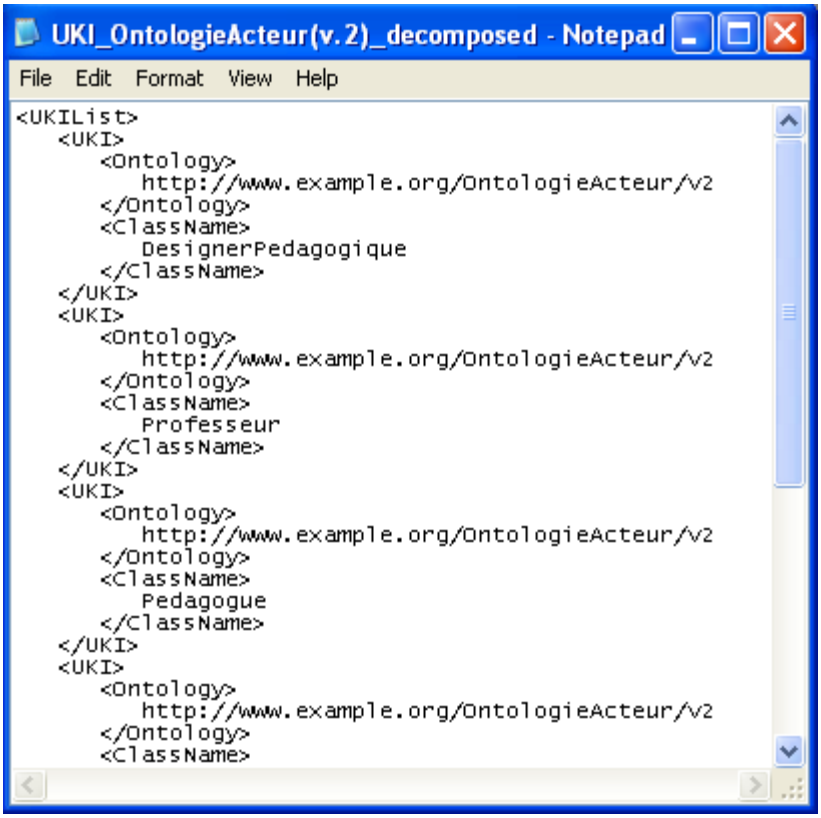


Figure V-8. Fichier des UKI décomposés en deux parties : l'URI de la (version) ontologie et le nom d'une classe utilisée comme référence sémantique

5.2.2.2. Extraire les changements apportés à la version V_N pour obtenir V_{N+1}

Pour extraire les changements apportés à la version V_N, SAM identifie d'abord l'URI de la version V_N à partir du fichier des UKI décomposés (p.ex. `http://www.example.org/OntologieActeur/v2`). Il vérifie également que tous les URI sont identiques au caractère près afin de s'assurer qu'ils se rapportent à la même version d'ontologie[46]. Ensuite, il envoie une demande d'identification des changements apportés à la version V_N pour obtenir V_{N+1} et le CHB lui fournit la version V_{N+1}Change avec la trace

[46] Bien qu'il existe aussi une autre situation, elle n'est pas supportée par le système SAM. Cette situation est celle où les URI, même s'ils ne sont pas identiques (caractère par caractère), peuvent se rapporter au même individu. Cependant, comparer et décider que deux URI différents identifient la même ressource induit un coût informatique élevé, ce pour quoi le W3C recommande d'éviter la pratique d'alias d'URI.

intégrée des changements. Étant donné que nous avons muni SAM de la capacité de lire et d'interpréter le langage OC (*OntologyChange*), ce système peut 'comprendre' les changements intégrés dans la V_{N+1}Change et peut donc extraire l'historique de l'évolution par lui-même.

5.2.2.3. **Mettre en correspondance les UKI avec les changements extraits**

Pour chaque UKI, le système SAM effectue une mise en correspondance entre le nom de classe et chacun des noms de classe qui apparaissent dans la trace de changements intégrée à V_{N+1}Change. Un exemple de mise en correspondance est fourni dans la Figure V-9, toutes les autres correspondances s'effectuant de la même manière.

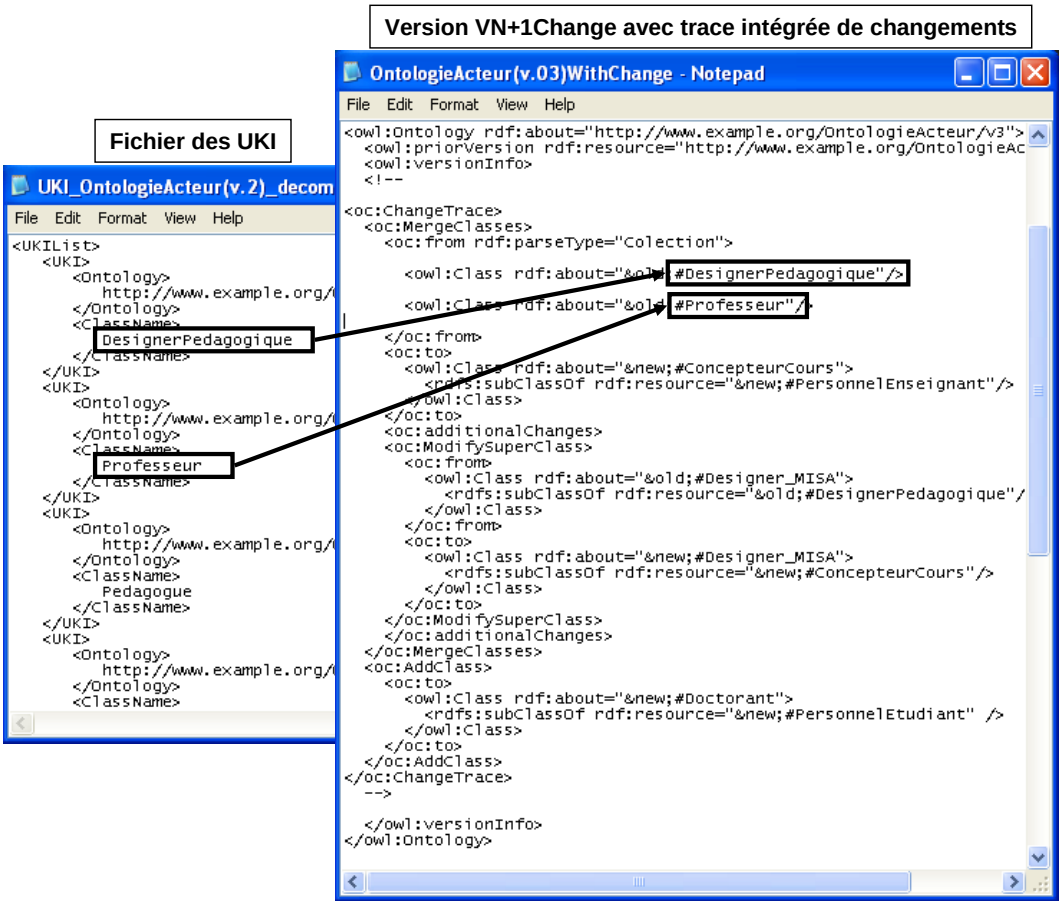


Figure V-9. Exemple de mise en correspondance des UKI avec les changements extraits

SAM collecte ensuite tous les changements pour lesquels des correspondances ont été trouvées et il génère un fichier contenant les correspondances, les changements collectés (ceux pour lesquels au moins une correspondance a été trouvée) et les UKI affectés.

5.2.3. **Analyser les changements**

Que les utilisateurs aient ou non envoyé des UKI, l'analyse des changements se réalise selon les mêmes lignes directrices, comme nous l'illustrons dans la Figure V-10. La seule différence réside dans le fait que SAM fournit une analyse générale de tous les changements, en présentant le pire des effets possibles, quand les UKI ne sont pas disponibles. Dans le cas contraire, il fournit une analyse plus contextuelle, axée uniquement sur les changements qui affectent les UKI. Les changements dont on parle ici sont ceux extraits lors de la procédure d'extraction des changements, procédure qui récupère les changements apportés à une certaine version V_N pour obtenir la version V_{N+1}.

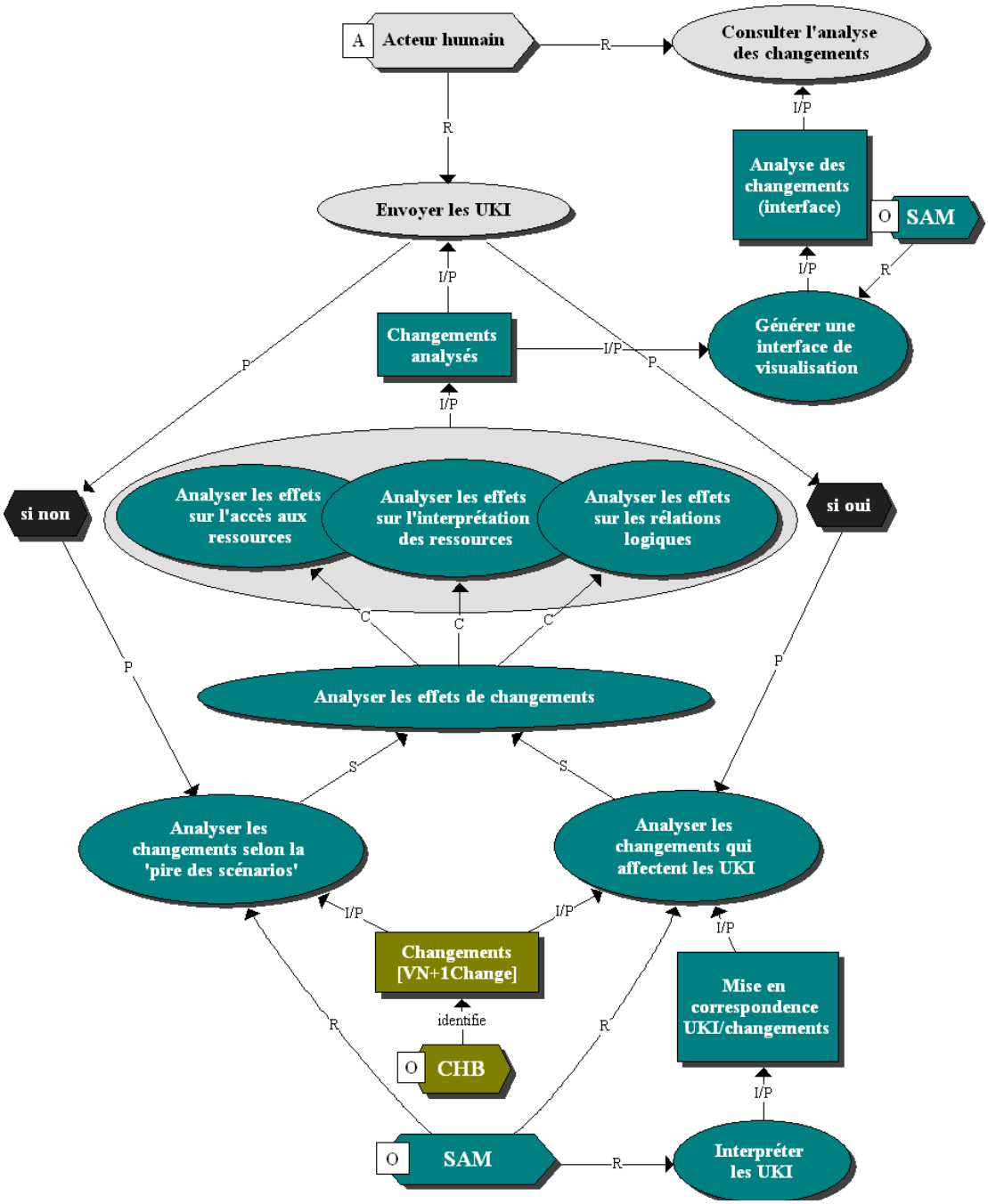


Figure V-10. Analyser les effets des changements avec SAM
(La légende du formalisme graphique est consultable dans l'Appendice A)

Les lignes directrices de l'analyse des effets[47] des changements sont au nombre de trois : (1) une identification des effets sur l'accès aux ressources référencées ; (2) une mise en évidence de la consistance ou de l'inconsistance de l'interprétation des ressources référencées ; (3) une découverte de relations logiques entre les classes appartenant à V_N et celles appartenant à V_{N+1}.

Dans le paragraphe suivant, nous allons donc expliquer ce que nous comprenons par accès aux ressources, par interprétation des ressources et finalement par relation logique entre les versions d'une même ontologie. Ensuite, nous allons illustrer à l'aide de certains exemples, l'analyse des effets des changements, telle qu'elle est réalisée par SAM. Finalement, nous montrons l'interface utilisée par le système pour présenter les résultats de l'analyse.

5.2.3.1. **Notions utilisées par SAM dans l'analyse des changements**

5.2.3.1.1. *Accès direct et indirect aux ressources*

Les ontologies fournissent un vocabulaire structuré servant de support aux requêtes permettant d'accéder aux ressources référencées sémantiquement par des classes spécifiées dans ces ontologies. L'accès aux ressources peut s'effectuer d'une manière directe ou indirecte (*cf*. Figure V-11).

[47] Voir aussi l'ontologie des changements, où nous avons spécifié les types d'effet des changements d'une manière plus formelle, en utilisant le langage de représentation OWL.

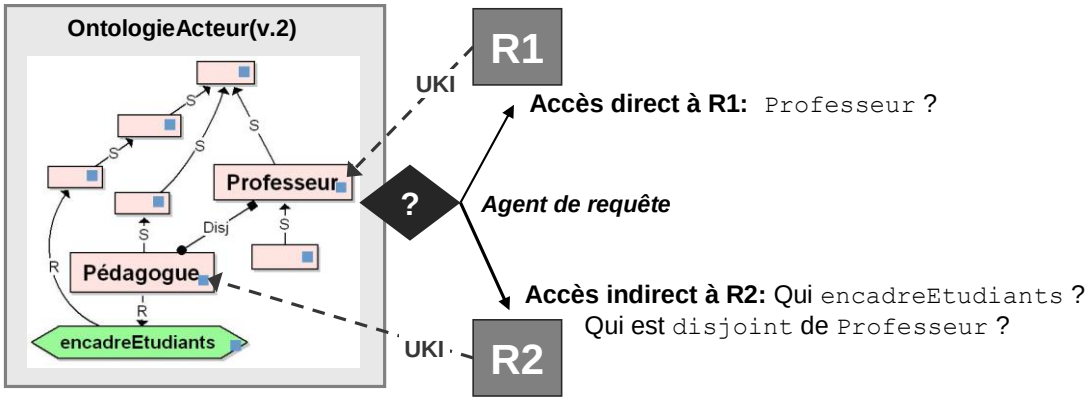


Figure V-11. Accès direct et indirect aux ressources référencées

> **Définition (Accès direct**). L'accès direct se réalise par des **requêtes** (ou recherches) **conceptuelles**, qui utilisent uniquement les classes définies dans une ou plusieurs ontologies afin d'obtenir les ressources référencées par ces classes.

Soit, par exemple, une ressource R1, référencée par une classe nommée `Professeur`. L'accès direct à la ressource R1 se réalise par des requêtes (ou recherches) qui utilisent la classe `Professeur` comme mot-clé. Un exemple d'une telle requête serait : « Donnez-moi toutes les ressources de type `Professeur` ».

> **Définition (Accès indirect).** L'accès indirect se réalise par des **requêtes** (ou recherches) **logiques**, qui utilisent des axiomes et des propriétés définies dans une ou plusieurs ontologies pour obtenir les ressources ayant les caractéristiques logiques spécifiées.

Soit, par exemple, une ressource R2, référencée par une classe nommée `Pédagogue` qui fait partie du domaine de la propriété `encadreEtudiants` et qui est disjointe de `Professeur`. L'accès indirect à la ressource R2 peut se réaliser par des requêtes qu'on appelle logiques (ou recherches logiques) puisqu'elles utilisent la logique spécifiée dans l'ontologie. Une requête de ce type serait : « Donnez-moi les ressources qui `encadreEtudiants` et qui ne sont pas `Professeur` ».

5.2.3.1.2. *Interprétation des ressources*

Les ressources sont référencées afin de décrire formellement leur contenu, leur utilisation ou le service qu'elles fournissent. Elles acquièrent ainsi une interprétation propre aux descripteurs sémantiques qui leur sont associés, c'est-à-dire propre aux UKI. Par exemple, dans la figure ci-dessus, l'interprétation de ce qu'est la ressource R2 est la suivante : « il s'agit d'un `Pédagogue` qui fournit un service d'encadrement aux étudiants (`encadreEtudiants`) mais qui n'est pas, pour autant, un `Professeur` ».

Définition (Interprétation des ressources). L'interprétation d'une ressource est la description formelle qui lui est associée au moyen des UKI.

5.2.3.1.3. *Relations logiques entre des versions d'une ontologie*

Étant donné que les ressources sont référencées par des classes provenant des ontologies, il est important de connaître l'effet des changements sur la signification des classes appartenant à V_N et à V_{N+1}. De cette manière, il est possible de proposer des solutions pour la modification du référencement des ressources et de préserver ainsi leur accès et leur interprétation via V_{N+1}, tel que nous le montrons plus loin dans ce chapitre.

Définition (Relations logiques entre versions d'ontologie). Les **relations logiques** désignent les relations entre une classe appartenant à V_N et la même ou une autre classe appartenant à V_{N+1} selon des critères comme : identité, équivalence, inclusion, généralisation, spécialisation, conceptuellement différentes ou même inexistence.

Par exemple, considérons la Figure V-12 qui présente une partie d'une version V_N (`OntologieActeur v.2`) et d'une version V_{N+1} (`OntologieActeur v.3`). Dans ce contexte, une relation logique est celle précisant que la classe `Pédagogue`, appartenant à la version V_{N+1}, est plus spécialisée que la même classe appartenant à la version V_N puisqu'on lui a ajouté une propriété qui la décrit de manière plus spécifique. Ou encore, la classe `ConcepteurCours` inclut la signification de `Professeur` et `DesignerPedagogique` puisqu'elle résulte de leur fusion en V_{N+1}.

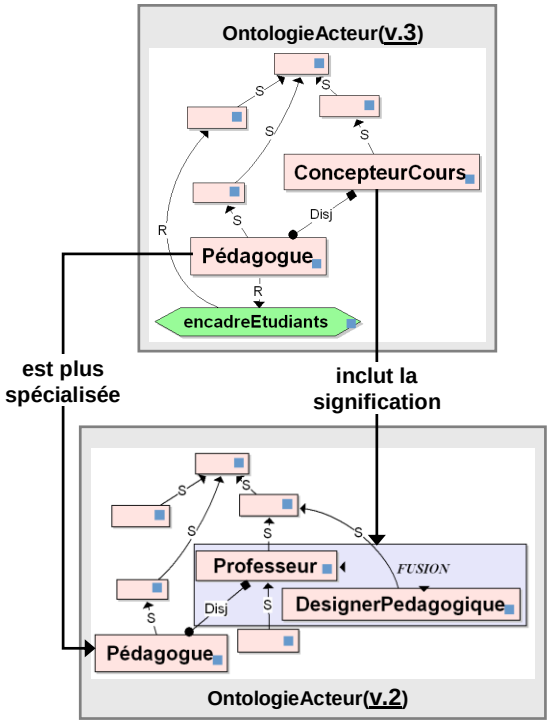


Figure V-12. Relations logiques entre V_N et V_{N+1}

5.2.3.2. **Analyse des effets en fonction du type de changement**

L'analyse de changements est implémentée dans SAM à l'aide d'un ensemble de règles qui, pour chaque changement, identifient des effets spécifiques. L'originalité de cette analyse réside dans l'étude conceptuelle sous-jacente et non dans l'implémentation informatique. C'est pourquoi nous présentons dans cette section uniquement les résultats de cette étude conceptuelle. Nous précisons aussi que cette analyse concerne uniquement les effets ponctuels, c'est-à-dire ceux qu'on obtiendrait si chaque changement était accompli indépendamment des autres.

5.2.3.2.1. *Analyse des changements de classes*

Les tableaux ci-dessous illustrent les effets des changements de classes, organisés selon les trois lignes directrices : (1) l'effet sur l'accès direct et indirect aux ressources ; (2) l'effet sur l'interprétation des ressources : (3) l'effet sur les relations logiques entre les

versions d'ontologie. Nous avons choisi de présenter ici uniquement quelques exemples : `DeleteClass`, `AddDisjointClass`, `ModifySuperClass` et `MergeClasses`.

Tableau V-1. Analyse du changement `DeleteClass`

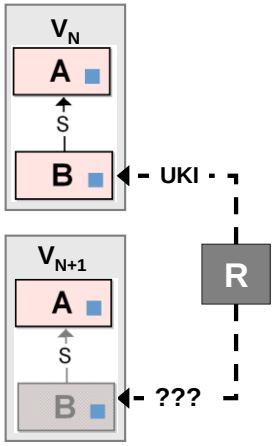
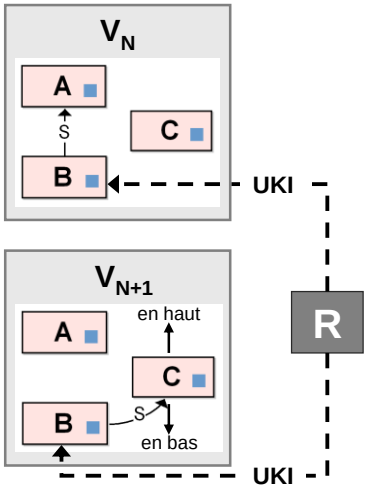
`DeleteClass` est un changement qui efface une classe d'une ontologie. (*exemple* : effacer la classe B, une sous-classe de A)	
	Effet sur le référencement sémantique **Effet sur l'accès direct via V_{N+1} :** Accès brisé, pour toute ressource référencée par la classe effacée (p.ex. la classe B). **Effet sur l'accès indirect via V_{N+1} :** S/O (sans objet). **Effet sur l'interprétation des ressources via V_{N+1} :** Interprétation diminuée, puisqu'une référence (UKI) n'est plus valable dans la description des ressources référencées aussi par la classe effacée.
	Relations logiques entre versions d'ontologie : La classe effacée n'existe plus en V_{N+1} (relation d'**inexistence**), mais sa superclasse (p.ex. la classe A) peut contenir sa signification à un niveau plus général.

Tableau V-2. Analyse du changement `ModifySuperClass`

`ModifySuperClass` est un changement qui modifie la superclasse d'une classe dans la hiérarchie. Il est équivalent avec le changement qui transfère une sous-classe d'une classe à une autre.

(*exemple* : pour la classe B, remplacer sa superclasse A par la classe C ou, en d'autres mots, transférer la sous-classe B de la classe A à la classe C)



Effet sur le référencement sémantique

Effet sur l'accès direct via V_{N+1} : Accès préservé, pour toute ressource référencée par la classe dont la superclasse a été modifiée (p.ex. la classe B).

Effet sur l'accès indirect via V_{N+1} : Accès modifié. La ressource n'est peut plus être accessible par des requêtes utilisant les axiomes et les propriétés de la superclasse modifiée (p.ex. la classe A). Elle peut être cependant accessible par des requêtes utilisant les axiomes et les propriétés de sa nouvelle superclasse (p.ex. la classe C)[48].

Effet sur l'interprétation des ressources via V_{N+1} : Interprétation consistante.

Relations logiques entre versions d'ontologie:

(1) Si la nouvelle superclasse est plus haut dans la hiérarchie, alors il se peut que des axiomes et des propriétés soient perdus pour la sous-classe transférée (p.ex. la classe B). Par conséquent, celle-ci devient plus générale (relation de **généralisation**) que sa correspondante en V_N.

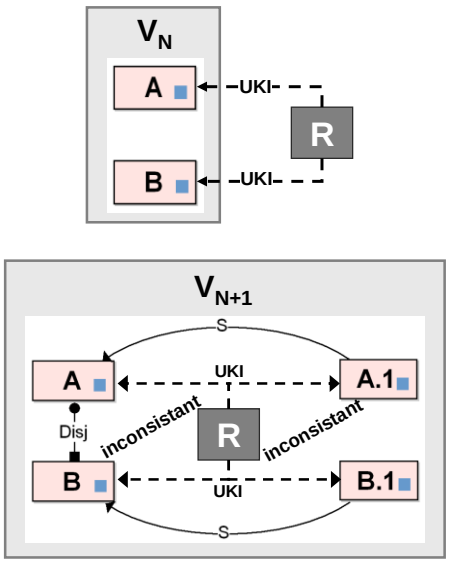
(2) Si la nouvelle superclasse est plus bas dans la hiérarchie, alors il se peut que des axiomes et des propriétés soient ajoutés à la sous-classe transférée. Par conséquent, celle-ci devient plus spécialisée (relation de **spécialisation**) que sa correspondante en V_N.

[48] Par exemple, considérons la classe `ConcepteurCours` qui appartient au domaine de la propriété `identifieConnaissances`. Considérons ensuite la classe `FacilitateurApprentissage`, qui appartient au domaine de la propriété `communiqueConnaissances`. Finalement, considérons la classe `Professeur` dont la superclasse `ConcepteurCours` a été modifiée par `FacilitateurApprentissage`. Dans ce cas, la réponse aux requêtes de type « donnez-moi les personnes qui peuvent identifier des connaissances » n'inclut plus les ressources référencées par la classe `Professeur`. Cependant, la réponse aux requêtes de type « donnez-moi les personnes qui sont capables de communiquer des connaissances » est étendue pour inclure aussi les ressources référencées par la classe `Professeur`.

Tableau V-3. Analyse du changement `AddDisjointClass`

`AddDisjointClass` est un changement qui déclare qu'une classe est disjointe d'une autre classe.

(*exemple* : ajout d'un axiome de disjonction entre les classes A et B)



Effet sur le référencement sémantique

Effet sur l'accès direct via V_{N+1} : Accès préservé, toutes les ressources référencées par les deux classes disjointes restent directement accessibles via V_{N+1}.

Effet sur l'accès indirect via V_{N+1} : Accès élargi, pour toute requête qui utilise le nouvel axiome. Un exemple d'une telle requête est « quelles sont les ressources disjointes de A ».

Effet sur l'interprétation des ressources via V_{N+1} : Interprétation inconsistante de ressources référencées par les deux classes déclarées disjointes en V_{N+1}, ou encore par leurs sous-classes.

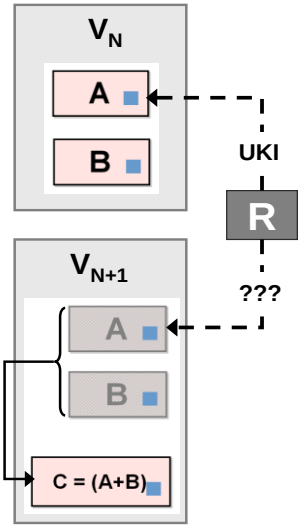
Relations logiques entre versions d'ontologie:

Les deux classes déclarées disjointes deviennent plus spécialisées en V_{N+1} qu'en V_N (relation de **spécification**) puisqu'on leur a ajouté des caractéristiques supplémentaires.

Tableau V-4. Analyse du changement `MergeClasses`

`MergeClasses` est un changement qui fusionne plusieurs classes dans une nouvelle classe.

(*exemple* : fusionner les classes A et B en une nouvelle classe C)



Effet sur le référencement sémantique

Effet sur l'accès direct via V_{N+1} : Accès brisé, pour toutes les ressources référencées par les classes fusionnées.

Effet sur l'accès indirect via V_{N+1} : S/O, les ressources ne sont plus directement accessibles *via* les classes fusionnées.

Effet sur l'interprétation des ressources via V_{N+1} : Interprétation diminuée, puisque des références (UKI) ne sont plus valables dans la description des ressources référencées par les classes fusionnées.

Relations logiques entre versions d'ontologie :

La classe résultante de la fusion, appartenant à la V_{N+1}, inclut la signification (relation d'**inclusion**) des classes fusionnées, appartenant à V_N.

5.2.3.2.2. *Analyse des changements de propriétés*

Comme pour les classes, les effets des changements de propriétés sont organisés selon leur impact sur l'accès et l'interprétation des ressources ou sur les relations logiques entre les versions d'ontologie. À titre d'exemple, nous présentons les changements `DeletePropertyDomain` et `ModifyPropertyDomain`.

Tableau V-5. Analyse du changement `DeletePropertyDomain`

`DeletePropertyDomain` est un changement qui efface une ou plusieurs classes du domaine d'une propriété d'objet ou de type de donnée (sans effacer complètement la classe de l'ontologie)	
	Effet sur le référencement sémantique **Effet sur l'accès direct via V_{N+1} :** Accès préservé. **Effet sur l'accès indirect via V_{N+1} :** Accès diminué, pour les requêtes utilisant la propriété d'où les classes ont été effacées[49]. **Effet sur l'interprétation des ressources via V_{N+1} :** L'interprétation reste consistante. Cependant, l'interprétation des ressources référencées par la ou les classes ayant été effacées du domaine de la propriété devient moins précise.
	Relations logiques entre versions d'ontologie : Les classes effacées du domaine de la propriété deviennent plus **générales** en V_{N+1} qu'elles étaient en V_N. Leur signification est moins expliquée.

Tableau V-6. Analyse du changement `ModifyPropertyDomain`

`ModifyPropertyDomain` est un changement qui modifie le domaine d'une propriété d'objet ou de type de donnée, en remplaçant une classe faisant partie du domaine avec une (ou plusieurs) autre classe.	
	Effet sur le référencement sémantique **Effet sur l'accès direct via V_{N+1} :** Accès préservé. **Effet sur l'accès indirect via V_{N+1} :** Accès modifié. Il est diminué pour les requêtes utilisant les classes d'où on a 'transféré' la propriété. Il est élargi pour les requêtes utilisant les classes auxquelles on a 'transféré' la propriété. **Effet sur l'interprétation des ressources via V_{N+1} :** L'interprétation reste consistante.

[49] Considérons la classe `ConcepteurCours` qui a été effacée du domaine de la propriété `identifieConnaissances`. La réponse à la requête « donnez-moi toutes les personnes qui sont capables d'identifier des connaissances » n'inclut plus les ressources référencées par la classe `ConcepteurCours`.

	Relations logiques entre versions d'ontologie : (1) Les classes ajoutées au domaine de la propriété deviennent plus **spécialisées** en V_{N+1}. Leur signification est mieux expliquée. (2) La classe effacée du domaine de la propriété devient plus **générale** en V_{N+1}. Sa signification est moins expliquée.

5.2.4. **Générer une interface de visualisation de l'analyse des changements**

Pour conclure, nous présentons l'interface du SAM pour la visualisation de l'analyse des effets des changements. Cette interface se compose de deux parties : la visualisation des changements problématiques et la visualisation de l'analyse reliée à chaque changement. Un exemple est fourni dans la Figure V-13.

Visualisation des changements problématiques

Le système SAM présente aux utilisateurs les changements apportés à V_N pour obtenir V_{N+1}. Pour cela il utilise la même interface de visualisation des changements que celle fournie par le CHB, mais il ajoute toutefois quelques particularités :

- Les changements qui brisent l'accès direct aux ressources sont soulignés en rouge;
- Les changements qui brisent ou modifient l'accès indirect sont soulignés en bleu, de même que ceux qui modifient l'interprétation d'une ressource, la diminue ou l'élargisse ;
- Les changements qui produisent une interprétation inconsistante des ressources sont soulignés en jaune.

Précisons que si les utilisateurs ont envoyé un fichier d'UKI, seuls les changements qui affectent les UKI sont mis en évidence, sinon tous les changements sont mis en évidence par SAM.

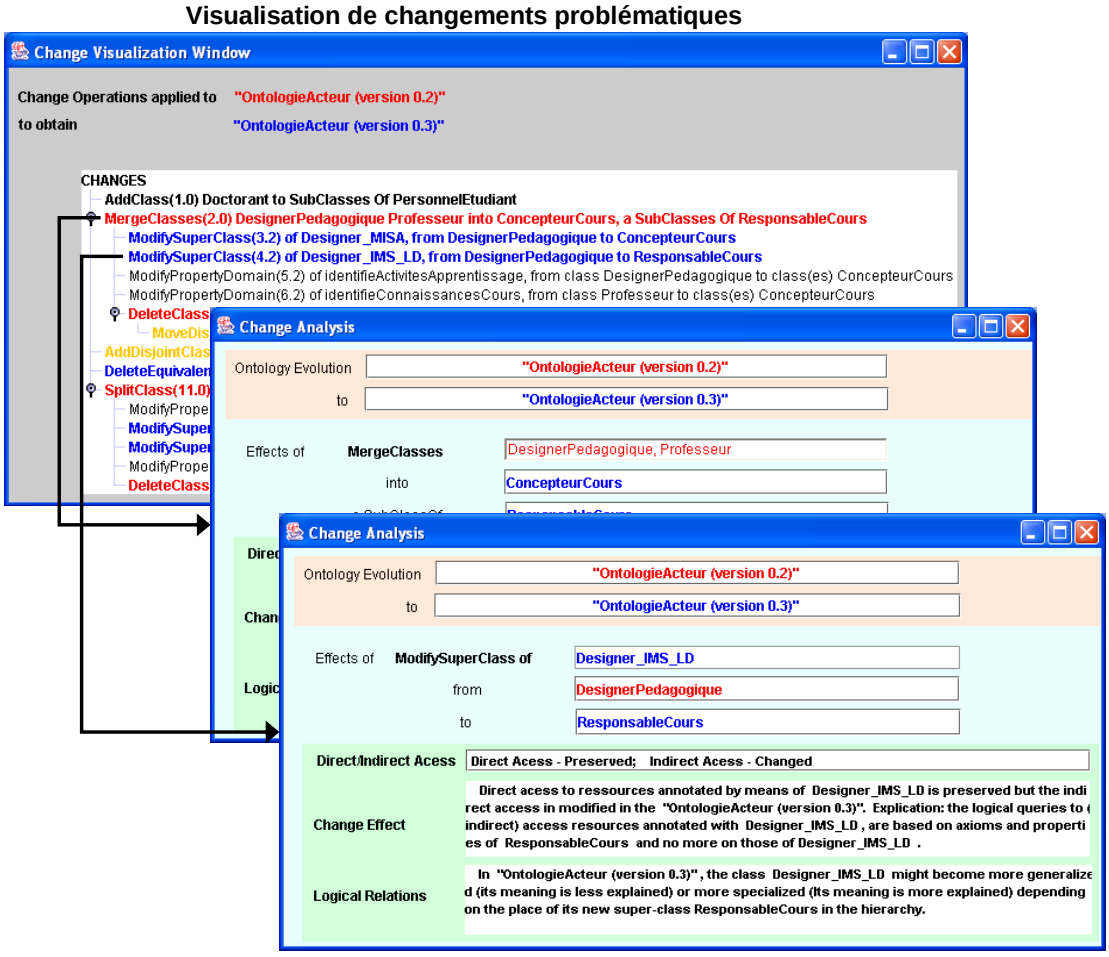


Figure V-13. Visualisation de changements problématiques et de leur analyse

Visualisation de l'analyse des effets des changements

Pour chaque changement souligné, une analyse des effets est fournie à la demande des utilisateurs (quand ils cliquent sur un changement dans la fenêtre de visualisation). Cette analyse comprend trois volets, dont les deux premiers traitent de l'effet du changement sur l'accès direct et l'accès indirect aux ressources référencées. Si le changement analysé a un effet négatif sur la consistance de l'interprétation de la ressource, alors cette information est également affichée dans ces deux volets. Le troisième volet désigne les relations logiques qui existent entre des classes appartenant à la V_N et à la V_{N+1}.

5.3 **Modification du référencement sémantique avec SAM**

Les changements apportés aux versions d'ontologie peuvent engendrer plusieurs effets sur le référencement sémantique des ressources. Parmi les changements les plus problématiques, on retrouve ceux qui brisent l'accès direct ou indirect ou ceux qui rendent inconsistante l'interprétation des ressources référencées. Après la fonctionnalité d'analyse des effets des changements, la deuxième fonctionnalité du système *SemanticAnnotationModifier* (**SAM**) concerne la modification du référencement sémantique pour préserver l'accès aux ressources ainsi que leur interprétation après l'évolution de l'ontologie de référence.

5.3.1. **Modification du référencement – fonctionnement du SAM**

Dans ce paragraphe, nous définissons ce qu'est la modification du référencement et nous montrons brièvement comment elle est supportée par SAM.

Définition (Référencement sémantique). Le référencement sémantique d'une ressource se compose d'un ou de plusieurs UKI qui réfèrent chacun à une classe unique appartenant à une ontologie (ou version d'ontologie) bien identifiée. À son tour, chaque UKI se compose de deux éléments : l'URI d'une ontologie (ou version) et le nom d'une classe appartenant à cette ontologie.

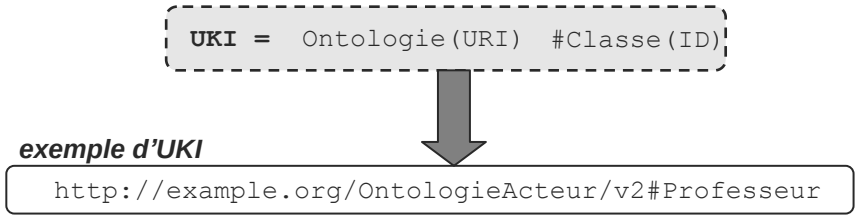


Figure V-14. Rappel de la définition d'un UKI (*cf.* Section 5.1.2.1.)

Définition (Modification du référencement sémantique). Le processus de transformation du référencement sémantique des ressources en fonction des changements apportés à une version d'ontologie V_N pour obtenir une nouvelle version V_{N+1}. Ce processus se traduit en réalité par la modification des UKI associés aux ressources. Le but est de permettre à ces dernières de référer à la version V_{N+1} d'une

manière consistante, c'est-à-dire d'une manière qui préserve l'accès et l'interprétation des ressources via V_{N+1}.

Dans la Figure V-15, nous illustrons brièvement la modification du référencement sémantique, tel qu'elle est supportée par le système SAM. Cette modification comporte trois étapes : l'organisation des UKI à modifier, la modification des UKI, la génération d'un fichier avec les UKI modifiés.

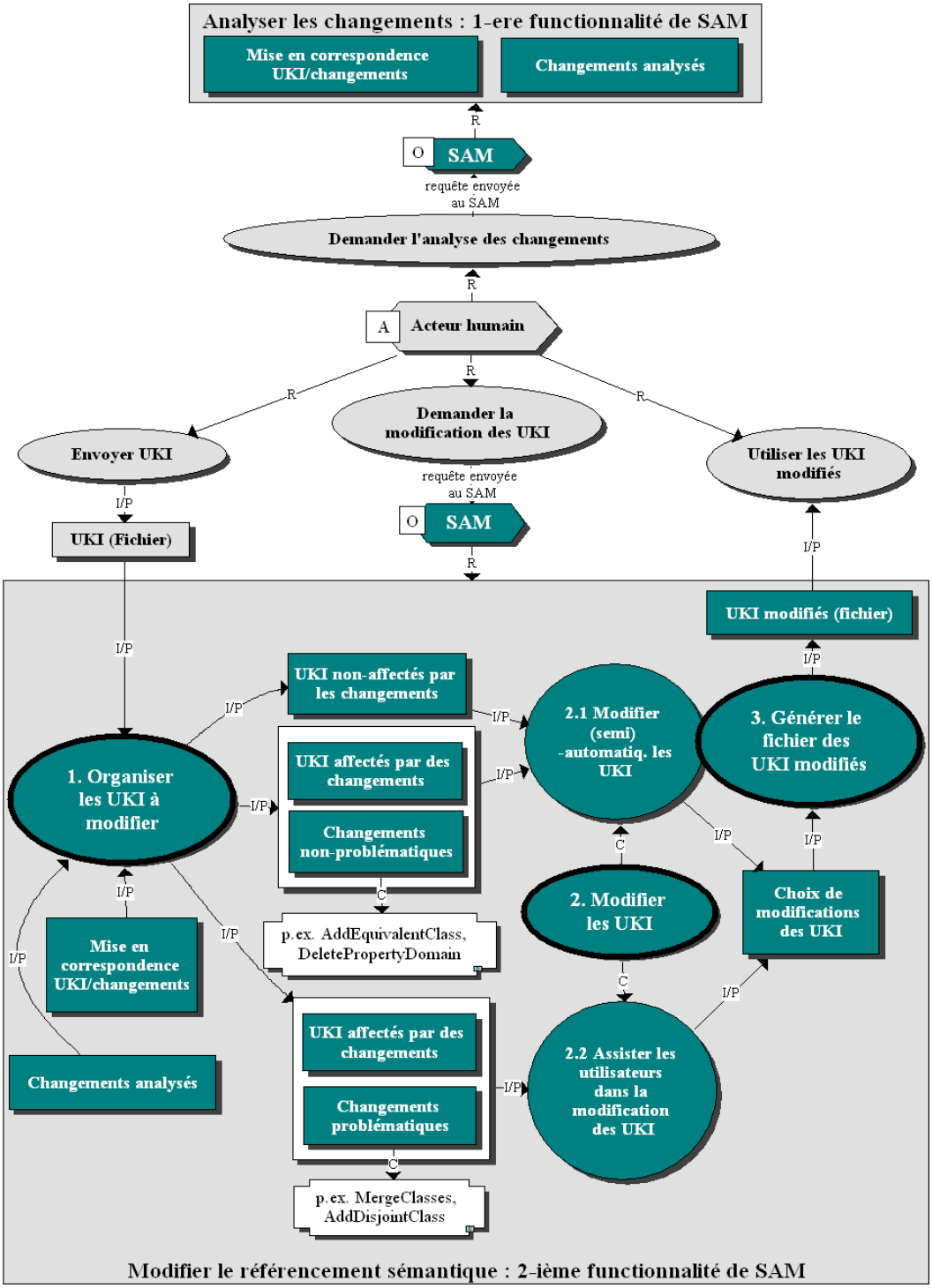


Figure V-15. Modification du référencement sémantique avec SAM

(La légende du formalisme graphique est consultable dans l'Appendice A)

L'organisation des UKI est une étape simple pendant laquelle le système SAM répartit les UKI à modifier en trois catégories : les UKI non affectés par les changements, ceux affectés par des changements n'ayant aucun effet problématique et, finalement, les UKI affectés par des changements ayant des effets problématiques, comme l'accès direct/indirect brisé/diminué ou encore une interprétation inconsistante.

La modification des UKI est une étape complexe, pendant laquelle les utilisateurs interagissent avec le système SAM qui les conseille et les guide lors de la modification du référencement. Nous observons alors que l'**acteur humain est toujours au centre du processus** : que ce soit pour des changements pas ou très problématiques, le système SAM fait toujours appel aux utilisateurs pour valider les modifications qu'il propose ou pour choisir parmi plusieurs modifications possibles. En fonction du type de changement, nous retrouvons cependant deux modalités de modification des UKI qui y sont affectés :

- *la modification (semi)automatique, avec l'accord des utilisateurs*. Cette modalité correspond aux deux premières catégories des UKI. Elle peut s'effectuer d'une manière plus automatique, en modifiant uniquement l'identifiant de la version de l'ontologie à l'intérieur de chaque UKI concerné. Cette modalité est présentée plus en détail dans le paragraphe 5.3.3.1.

- *la modification assistée par SAM, impliquant activement les utilisateurs*. Cette modalité correspond à la troisième catégorie des UKI, ceux qui sont affectés par des changements problématiques menant à une perte d'accès aux ressources ou encore à une interprétation inconsistante. Dans ce cas, il faudrait modifier, en plus de l'identifiant de la version, l'identifiant de la classe qui sert de référence sémantique à l'intérieur de chaque UKI affecté. Étant donné qu'elle se situe à un niveau sémantique riche, la modification des identifiants de classes peut difficilement être automatisable, et cela, même si on connaît les changements apportés à l'ontologie. C'est pourquoi il serait plus intéressant de guider les utilisateurs lors de la modification des UKI affectés par des changements problématiques, en leur proposant différentes solutions possibles et en leur donnant la possibilité de choisir parmi celles qui leur semblent les mieux adaptées à leur contexte. Cette modalité est présentée plus en détail dans le paragraphe 5.3.3.2.

La génération des UKI est une étape simple, pendant laquelle le système SAM fournit aux utilisateurs le fichier final avec les UKI modifiés selon un mode (semi)automatique ou assisté.

5.3.2. **Organiser les UKI à modifier**

Pour permettre la modification du référencement sémantique, chaque utilisateur rend disponible son propre fichier des UKI. Tel que discuté auparavant, SAM accède à ce fichier et l'interprète en décomposant chaque UKI dans ses deux parties constitutives (l'URI de l'ontologie et le nom de classe) et en mettant en correspondance les UKI avec les changements. Ensuite, le système SAM répartit les UKI à modifier en trois catégories. La première catégorie concerne les UKI qui ne sont pas affectés par les changements. La deuxième catégorie concerne les UKI qui sont affectés par des changements qui ne provoquent ni une perte d'accès aux ressources, ni une interprétation inconsistante. Un exemple de ce type de changements est `AddEquivalentClass`, qui ajoute un axiome d'équivalence à une classe, ou encore `ModifySuperClass`, qui modifie la superclasse d'une classe dans la taxonomie. La troisième catégorie concerne les UKI affectés par des changements problématiques, des changements comme le `DeleteClass` ou `MergeClasses` qui produisent une perte d'accès aux ressources référencées, ou encore `AddDisjointClass` qui peut engendrer une interprétation inconsistante.

5.3.3. **Modifier les UKI avec SAM**

Dans cette section nous décrivons les deux modalités de modification du référencement sémantique : la modification (semi)automatique et celle assistée.

5.3.3.1. **Modifier (semi)automatiquement les UKI**

Dans la Figure V-16, nous illustrons la modification (semi)automatique des UKI, tel qu'elle est supportée par SAM.

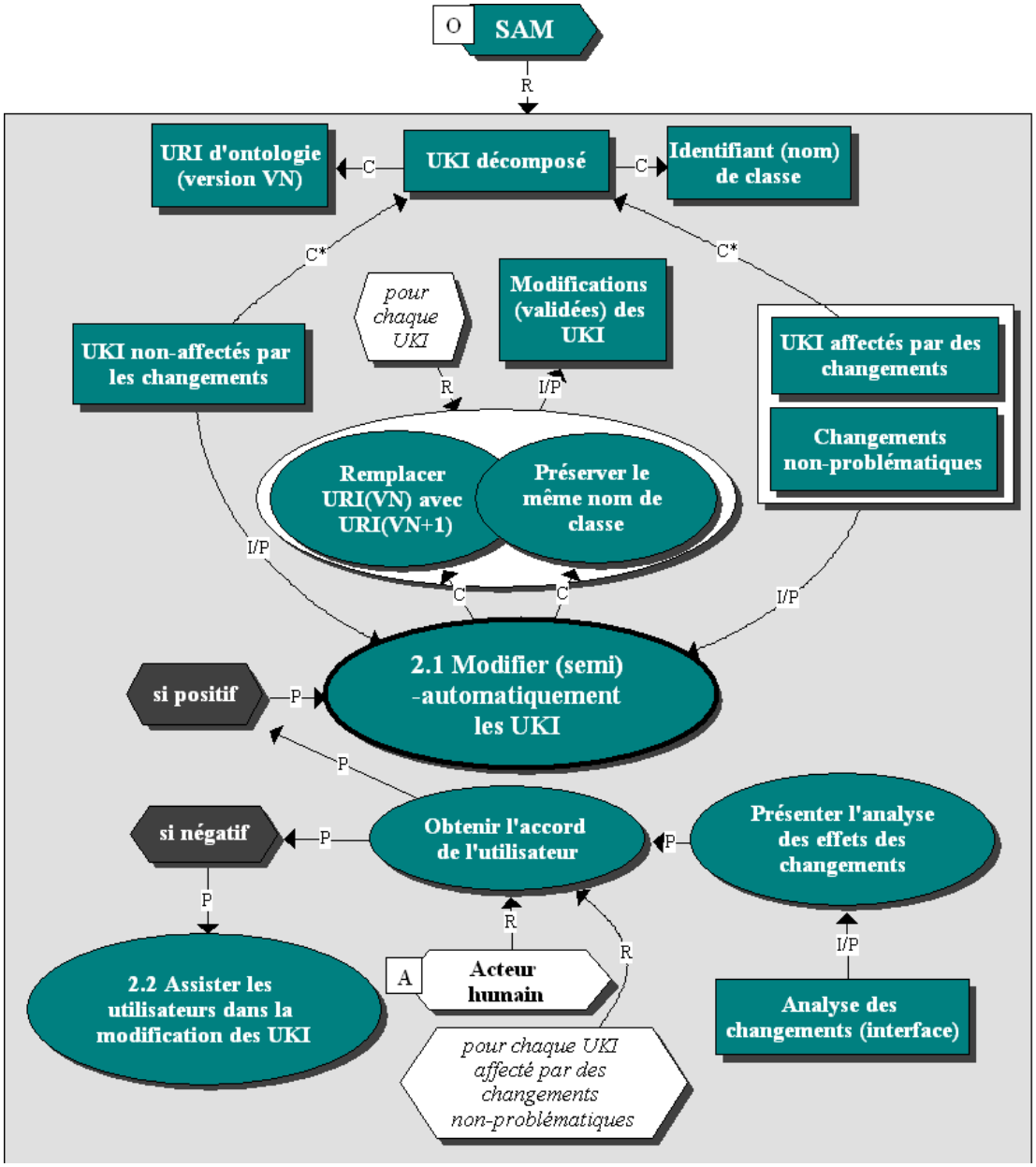


Figure V-16. Modification (semi)automatique des UKI non-affectés ou affectés par des changements non problématiques

(La légende du formalisme graphique est consultable dans l'Appendice A)

Premièrement, le système SAM procède à la modification automatique des UKI n'ayant aucune correspondance[50] avec les changements apportés pour passer d'une version d'ontologie V_N à une version V_{N+1}. Étant donné que les classes utilisées dans les UKI ne sont affectées par aucun des changements apportés à l'ontologie, leur signification n'est pas modifiée. Le référencement préserve donc son entière validité. Dans ce cas, il faudrait modifier uniquement l'URI de la version à l'intérieur des UKI, afin de permettre à ceux-ci de référer à la nouvelle version de l'ontologie, comme exemplifié dans la Figure V-17 (l'exemple de la ressource `R1` qui réfère à la classe `ChercheurEnseignant`). Cette modification ne nécessite pas une validation de la part des utilisateurs, mais ils seront toutefois informés à ce sujet.

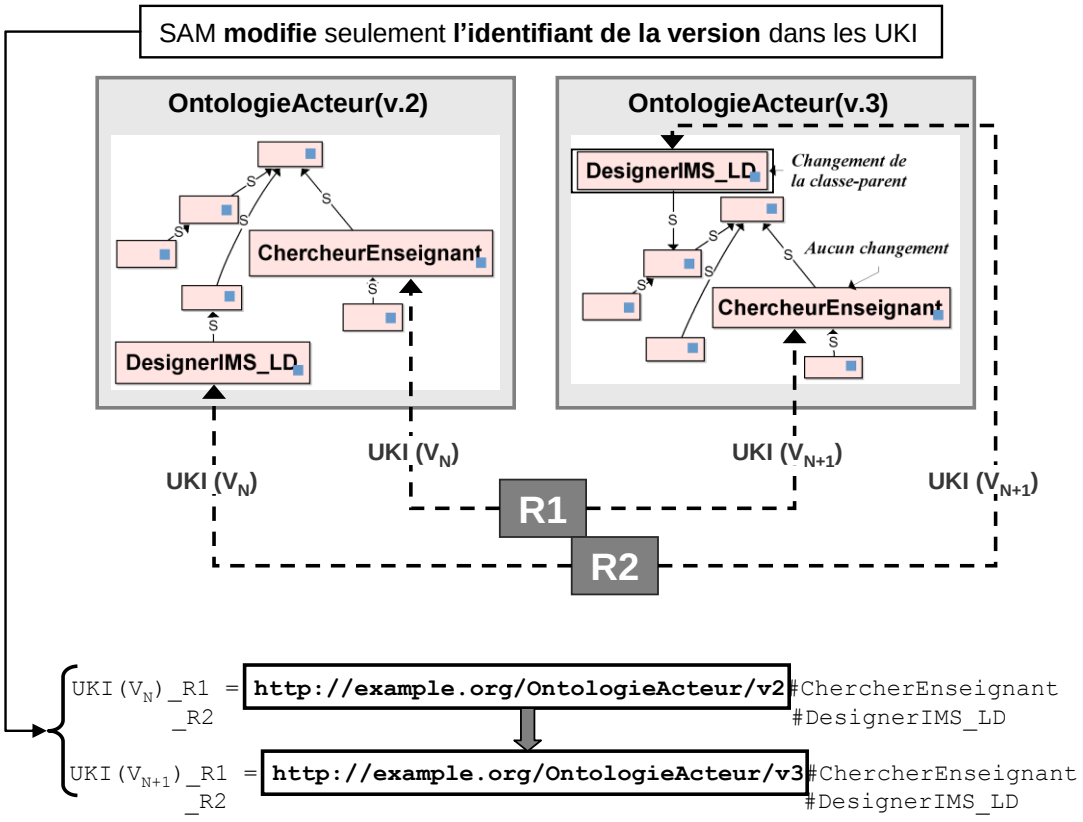


Figure V-17. Modification (semi)automatique – quelques exemples

[50] C'est-à-dire qu'aucune correspondance n'a été trouvée entre le nom d'une classe, spécifiée par un UKI, et les changements apportés.

Deuxièmement, le système SAM procède à la modification (semi)automatique des UKI affectés par des changements non problématiques. Le terme (semi)automatique signifie que les utilisateurs doivent valider chaque modification, même si elle ne concerne que le remplacement de l'URI de la version V_N avec celui de la version V_{N+1}, comme exemplifié dans la Figure V-16. Prenons l'exemple de la ressource `R2` qui réfère à la classe `DesignerIMS_LD`. Même si la classe `DesignerIMS_LD` reste valide en V_{N+1}, sa signification est modifiée puisqu'elle a été transférée à une autre classe dont elle hérite les propriétés : `DesignerIMS_LD`, comme sous-classe de `DesignerPedagogique` n'est pas exactement la même chose que `DesignerIMS_LD`, comme sous-classe de `PersonnelEnseignant`. Dans ce cas, pour chaque UKI affecté par des changements non-problématiques, il est nécessaire de montrer à l'utilisateur l'analyse des changements en question (p.ex. l'analyse de `ModifySuperClass`) et de lui demander s'il est d'accord de modifier uniquement l'identifiant de la version d'ontologie. Si l'utilisateur veut également modifier l'identifiant de classe dans l'UKI, il est redirigé vers le mode de modification assistée.

5.3.3.2. **Assister les utilisateurs pendant la modification des UKI affectés par des changements problématiques**

Ce paragraphe présente la modification des UKI affectés par des changements problématiques, menant à une perte d'accès ou à une interprétation inconsistante des ressources référencées. Dans ce cas, la plupart des classes utilisées dans les UKI ne sont plus disponibles dans la V_{N+1} (p.ex. l'effet engendré par `DeleteClass` ou `MergeClasses`). Ou, s'il existe des classes disponibles, il se peut que leur nouvelle signification crée des problèmes d'interprétation des ressources référencées (p.ex. l'effet engendré par le changement `AddDisjointClass`). Il faudrait alors modifier, en plus de l'URI de la version d'ontologie, les noms de classe dans les UKI affectés, afin de rendre ces derniers compatibles avec la nouvelle version d'ontologie. Un exemple est fourni dans la Figure V-18 où, pour un changement de type `MergeClasse`, l'UKI de la ressource `R` est modifié de la manière suivante :

- l'URI de la version V_N et remplacé avec celui de la version V_{N+1};

- le nom de classe `DesignerPedagogique` est remplacé par celui de `ConcepteurCours`, qui réfère à une classe appartenant à la V_{N+1}, spécifiquement à la classe résultante de la fusion entre `Professeur` et `DesignerPedagogique`.

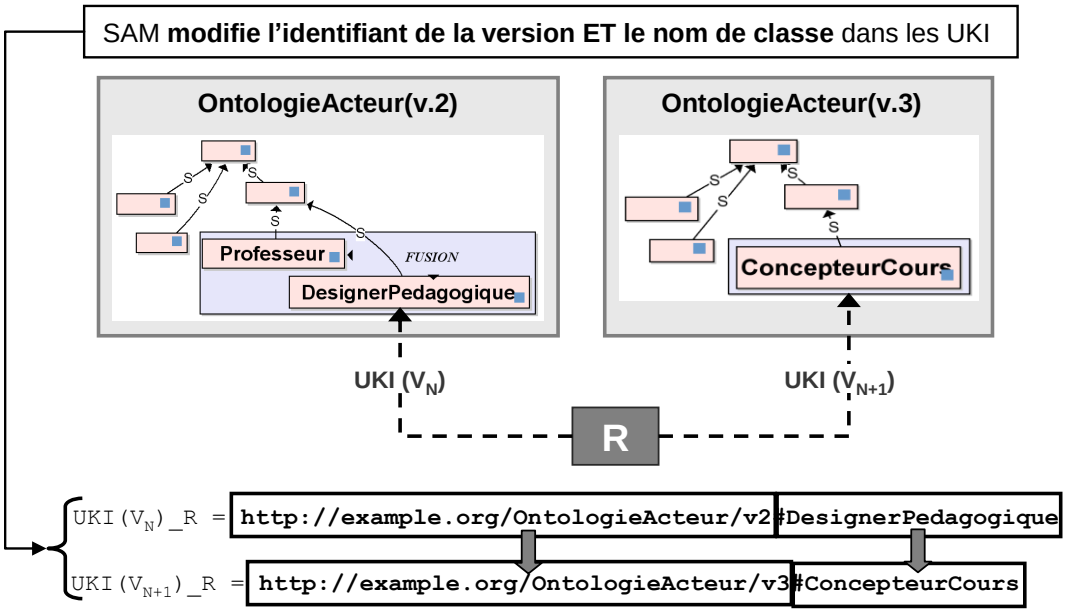


Figure V-18. Modification des UKI affectés par des changements problématiques – un exemple

Cette modification ne peut cependant pas se réaliser d'une manière automatique. Considérons, par exemple, un changement qui efface une classe `C1`, sous-classe de `C`. En faisant l'hypothèse que la superclasse `C` englobe la signification de la classe effacée, il est possible de modifier l'UKI pour y référer. Toutefois, d'autres possibilités sont également valables, comme celle de modifier l'UKI pour qu'il réfère à une des sous-classes de `C1` et non à sa superclasse. Faire un choix parmi l'ensemble de modifications possibles reste l'apanage des utilisateurs, ce sont eux qui peuvent juger de la pertinence sémantique d'une modification, ce sont eux qui peuvent savoir quelle modification serait la plus adéquate dans leur contexte. Bien que SAM puisse les aider, leur fournir des suggestions, les guider dans leurs tâches, le choix final de modification des UKI reste celui des utilisateurs.

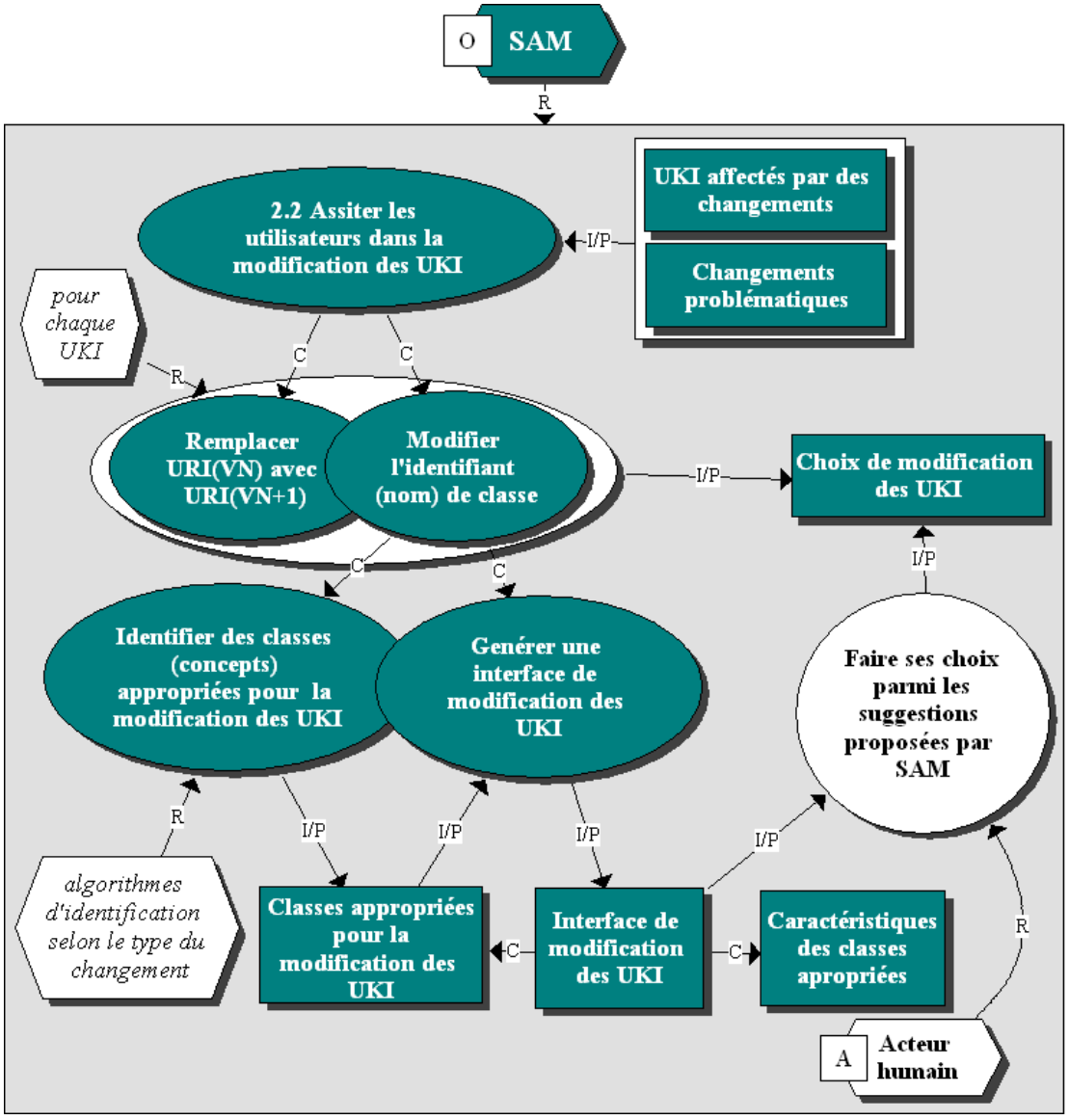


Figure V-19. Modification assistée par SAM pour les UKI affectés par des changements problématiques

(La légende du formalisme graphique est consultable dans l'Appendice A)

Dans la Figure V-19, nous montrons brièvement de quelle manière SAM assiste les utilisateurs lors de la modification des UKI affectés par des changements problématiques, particulièrement lors de la modification de l'identifiant (nom) de classe à l'intérieur des UKI. Pour chaque UKI affecté, SAM identifie premièrement les classes les plus appropriées à la modification en utilisant des algorithmes d'identification que nous avons développés. Ensuite

il génère une interface lui permettant d'interagir avec les utilisateurs et de les guider lors du processus de modification des UKI. Cette interface présente, en plus des classes identifiées comme étant appropriées à la modification des UKI, des caractéristiques propres à chaque classe.

5.3.3.2.1. *Identification des classes appropriées à la modification des UKI*

Dans ce paragraphe, nous discutons le raisonnement qu'utilise SAM pour détecter les classes appropriées à la modification des UKI affectés par des changements problématiques. Nous commençons par illustrer ce raisonnement, en l'appliquant au cas du changement `MergeClasses` (nous précisons que le système SAM applique le même raisonnement pour tout autre type de changement problématique, comme `DeleteClass` ou `SplitClass`). Nous terminons par présenter les algorithmes utilisés par SAM lors du processus de détection des classes.

Identification des classes appropriées à la modification des UKI – le cas du `MergeClasses`

La Figure V-20 présente l'exemple du changement qui fusionne deux classes (`Professeur` et `DesignerPedagogique`), appartenant à la version V_N, dans une nouvelle classe (`ConcepteurCours`), appartenant à la V_{N+1}. Soit aussi les ressources R_1 et R_2 ayant des UKI qui référent aux classes fusionnées. Ces UKI doivent être modifiés pour qu'ils référent à d'autres classes appartenant à V_{N+1}.

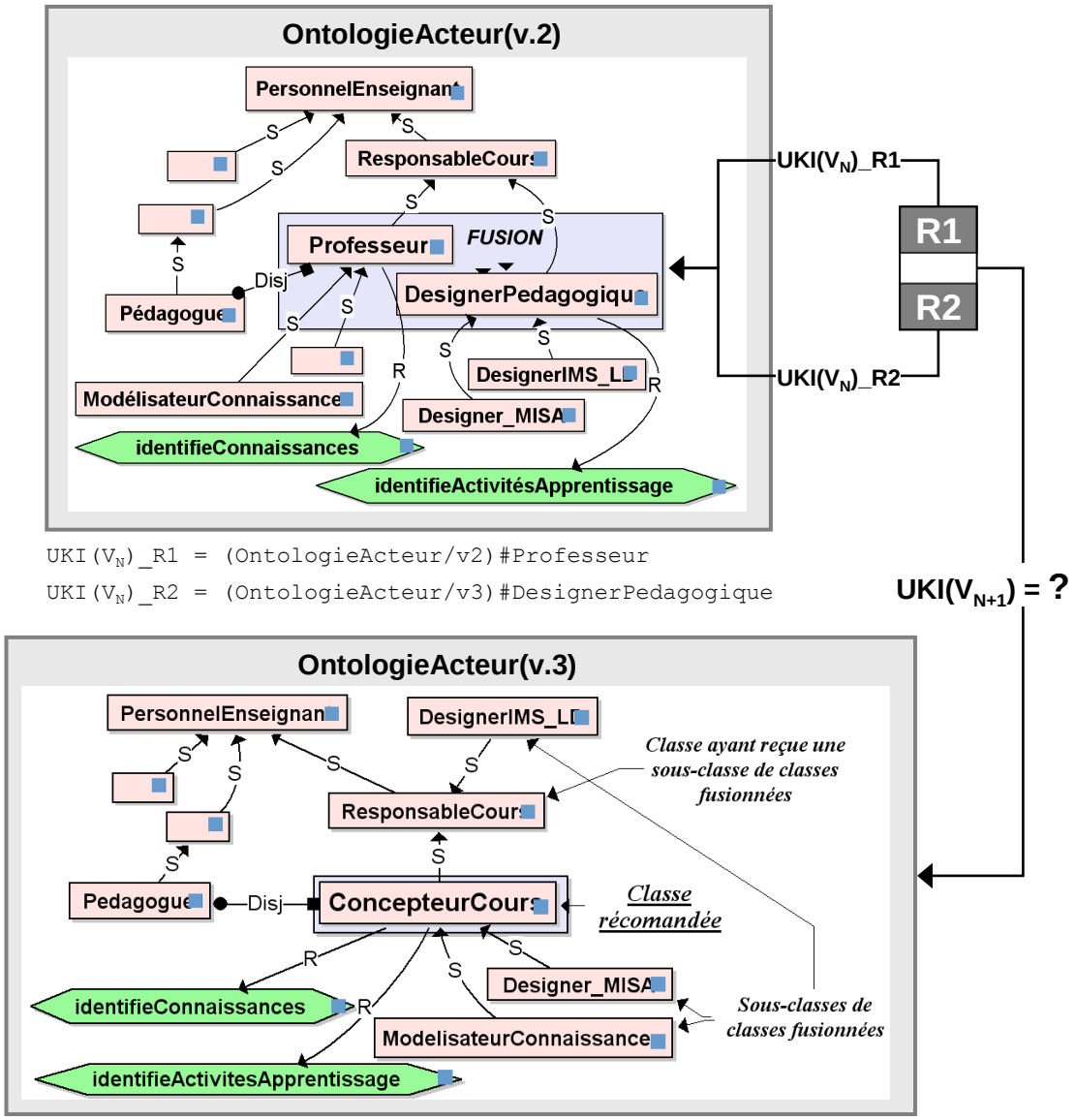


Figure V-20. Identification des classes pertinentes pour la modification des UKI – l'exemple du `MergeClasses`.

Cette figure met aussi en évidence les classes identifiées par SAM comme étant pertinentes à la modification des UKI affectés par le changement `MergeClasses`. Ces classes sont regroupées en trois catégories d'un degré de pertinence décroissant :

- les classes *proches sémantiquement*, c'est-à-dire les classes dont la signification est rapprochée de celles dont le nom doit être remplacé dans les UKI affectés. C'est la catégorie la plus recommandée.

  Dans le cas du changement `MergeClasses`, on peut supposer que la classe résultante, nommée `ConcepteurCours`, est proche sémantiquement des classes `Professeur` et `DesignerPedagogique` puisqu'elle inclut leur signification, bien qu'à un niveau plus général.

- les *sous-classes directes* des classes dont le nom doit être remplacé dans les UKI affectés. Il s'agit ici des sous-classes de premier niveau qui n'ont pas été effacées lors de la disparition de leur classe-parent, mais qui ont été transférées à d'autres classes en V_{N+1}. Par exemple, les trois sous-classes `ModelisateurConnaissances`, `DesignerIMS_LD`, `Designer_MISA` qui ont été transférées lors de la fusion de leurs classes-parent.

- les classes ayant d'autres rapprochements avec la classe dont le nom doit être remplacé dans les UKI affectés. Pour le moment, SAM identifie uniquement les classes auxquelles des sous-classes directes ont été transférées, comme la classe `ResponsableCours` qui a reçu une des sous-classes de classes fusionnées. Une question, que nous n'avons pas traitée dans cette thèse, mais qui fait partie des perspectives identifiées, est de voir s'il n'y a pas d'autres classes pertinentes pour la modification des UKI (p.ex. les classes auxquelles on a transféré un certain nombre de propriétés ou d'axiomes et qui ne font pas partie des classes déjà identifiées).

Algorithmes d'identification des classes appropriées à la modification des UKI – le cas du `MergeClasses`

Pour détecter les classes appropriées à la modification des UKI affectés par des changements problématiques, le système SAM utilise deux types d'algorithmes que nous avons développés (*cf.* Figure V-21). Ces algorithmes sont fondés sur la trace des changements intégrés dans la V_{N+1}Change, trace qui décrit les changements apportés à V_N pour obtenir V_{N+1}. Pour chaque changement problématique, le premier algorithme (nommé

AdditionalChanges) interprète ses changements additionnels et les regroupe dans différents ensembles (p.ex. l'ensemble des sous-classes transférées ou celui des propriétés effacées). C'est un algorithme commun à tous les changements. Le deuxième algorithme (nommé **ClassesIdentification**) est propre à chaque changement problématique. Il utilise les résultats fournis par le premier algorithme ainsi qu'un ensemble prédéfini de règles afin d'identifier les classes le plus appropriées à la modification des UKI affectés.

```
FOR each Change_I∈[Change_1…Change_N]=list of problematic changes that
affect UKI, DO

  IF Change_I = DeleteClass(C)
     Algorithm.AdditionalChanges
     Algorithm.ClassesIdentification(DeleteClass)
  EndIf

  IF Change_I = MergeClassses(C1 … CN) into C

  --- extraction des informations reliées aux changements additionnels ---
     Algorithm.AdditionalChanges
        Input: additional changes of the Change_I
        Output: organized information about additional changes

  --- identification des classes appropriées à la modification des UKI ---
     Algorithm.ClassesIdentification(MergeClasses)
        Input: results of Algorithm.AdditionalChanges
        Output: list of pertinent classes for the UKI modification
  EndIF

  IF Change_I = SplitClass C into (C1 … CN)
     Algorithm.AdditionalChanges
     Algorithm.ClassesIdentification(SplitClass)
  EndIF

  --- autres changements ---
EndFOR
```

Figure V-21. Algorithmes d'identifications des classes appropriées à la modification des UKI

Dans le cas du changement `MergeClasses`, les algorithmes prennent la forme illustrée dans la Figure V-22. Nous avons choisi de donner un seul exemple, puisque tout autre changement problématique sera traité d'une manière similaire.

If $Change_I$ = MergeClasses(C_1…C_N) into C

Algorithm.AdditionalChanges

BEGIN

FOR each $AdCh_I \in [AdCh_1 … AdCh_N]$= list of additional changes of $Change_I$, **DO**

--- création des ensembles reliées aux sous-classes effacées/ajoutées/transférées ---

IF $AdCh_I$=DeleteClass(X_1) **THEN** add the couple(X_1; X_2) to DeteledSubClasses set, **EndIF;**

IF $AdCh_I$=AddClass(Y_1) **THEN** add the couple (Y_1; Y_2) to AddedSubClasses set, **EndIF;**

IF $AdCh_I$=ModifySuperClass(X_1=Y_1) **THEN** add the triple (X_1=Y_1; X_2 ;Y_2) to TransferredSubClasses set, **EndIF;**

--- création des ensembles reliées aux axiomes effacés/ajoutés/transférés ---

(raisonnement identique à la fonction ci-dessus)

--- création des ensembles reliées aux propriétés effacées/ajoutées/transférées ---

(raisonnement identique à la fonction ci-dessus)

EndFOR

END

Algorithm.ClassesIdentification

BEGIN

RecommendedClass = C (*--- classe proche sémantiquement ---);*

FOR each triple $T_I \in$ TransferredSubClasses set, **DO**

IF $X_2 \in (C_1 … C_N)$ **THEN**

--- identification des sous-classes directes ---

add (X_1=Y_1) to the set SubClasses($Change_I$);

--- identification de classes ayant reçu des sous-classes de classes fusionnées ---

IF Y2≠C **THEN** add Y_2 to the set ClassesWithDirectSousClasses($Change_I$), **EndIF**

EndIF

EndFOR

END

Figure V-22. Algorithmes d'identification des classes appropriées à la modification des UKI – l'exemple de `MergeClasses`

(la signification des variables X et Y a été donnée au paragraphe 4.2.1.2.2.)

5.3.3.2.2. *Interaction entre SAM et les utilisateurs pour la modification des UKI*

La modification des UKI affectés par des changements problématiques est effectuée par les utilisateurs à l'aide du système SAM qui leur offre des suggestions concernant les modifications les plus pertinentes ainsi qu'une aide au niveau de l'interface. Dans ce

paragraphe, nous décrivons l'interaction entre les utilisateurs et le système SAM lors de la modification des UKI. Nous prenons comme exemple le changement `MergeClasses` qui affecte les UKI attachés respectivement aux ressources R_1 et R_2 (*cf.* Figure V-20).

L'utilisateur demande la modification des UKI

Pour demander la modification des UKI, l'utilisateur consulte d'abord la liste des changements apportés à V_N pour obtenir V_{N+1}, où les changements problématiques qui affectent les UKI sont soulignés en rouge (ou bleu, pour ceux moins problématiques). Ensuite, pour chaque changement problématique, il consulte l'analyse des changements. L'utilisateur décide s'il désire modifier les UKI qui en sont affectés, en cliquant sur le bouton *ModifyUKI* qui se trouve en bas de la fenêtre d'analyse (*cf.* Figure V-23). L'interface de modification est alors générée par le système SAM.

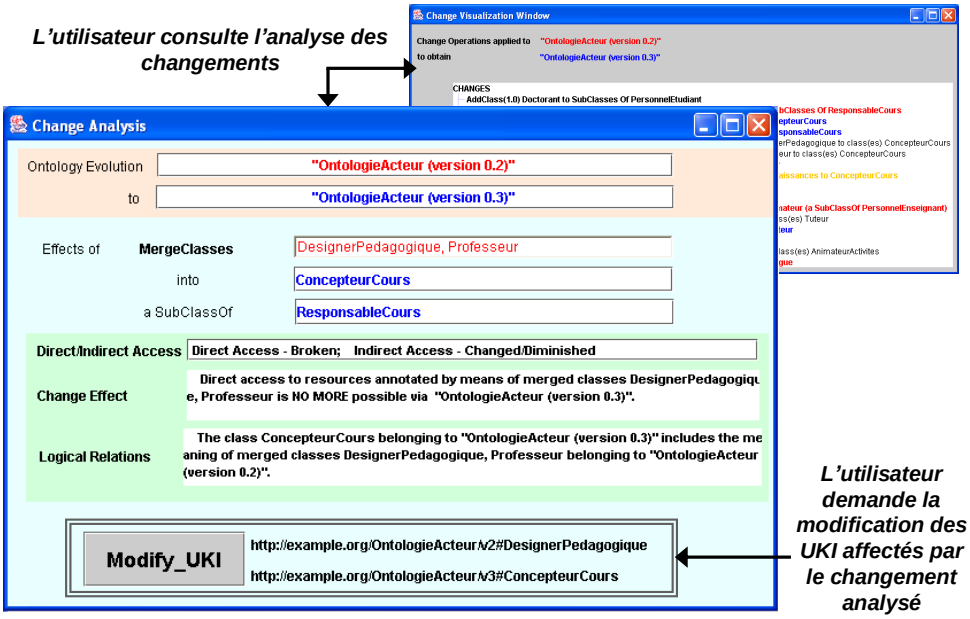


Figure V-23. Demande de modification des UKI

Le système SAM fournit des suggestions pour la modification des UKI

À la demande de l'utilisateur, le système SAM lui fournit des suggestions et des conseils pour la modification des UKI. Ces suggestions et conseils sont rassemblés dans une seule interface, que nous présentons dans la Figure V-24.

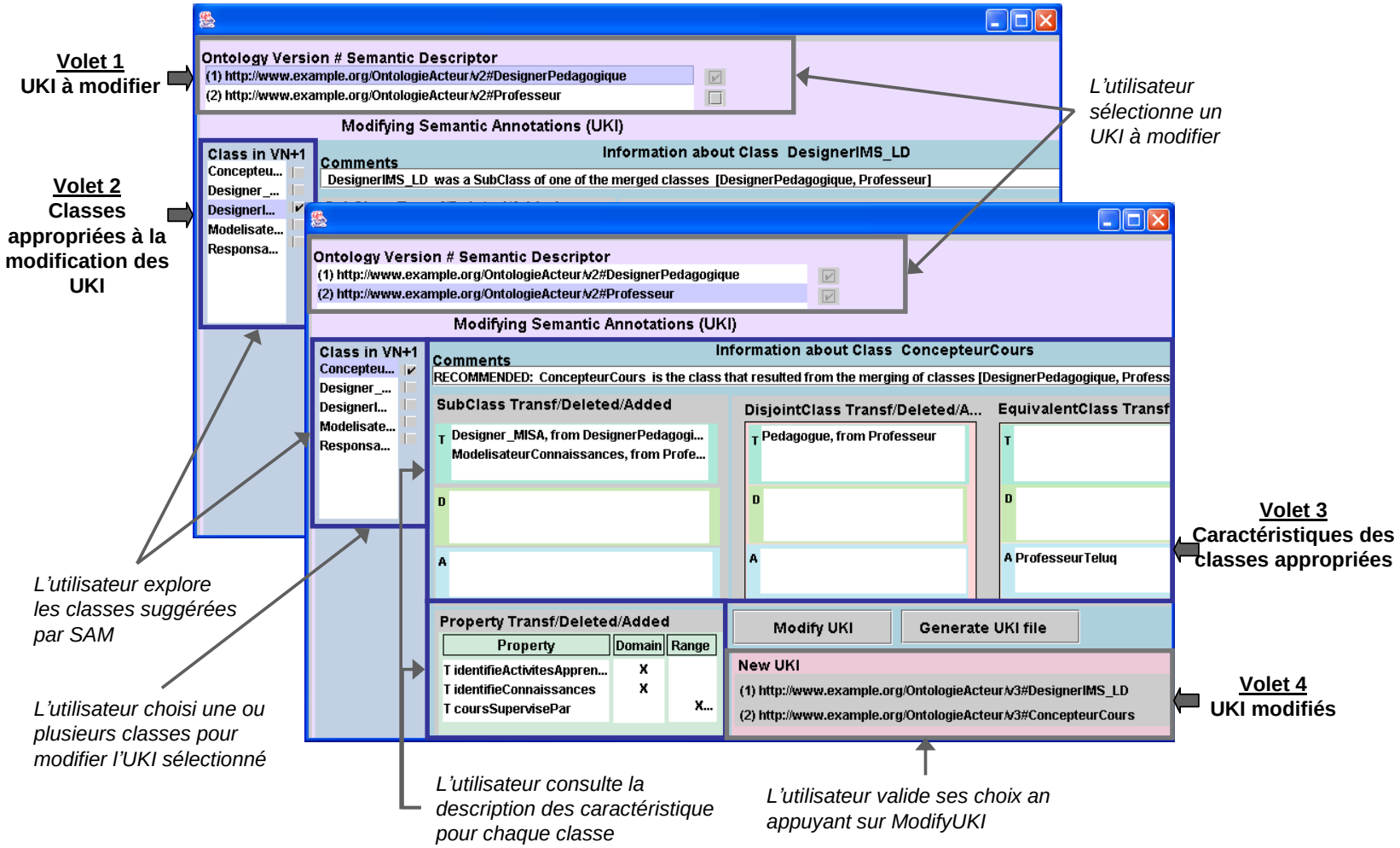


Figure V-24. Interface de SAM pour la modification des UKI.

Comme montré dans la figure ci-dessus, nous pouvons décomposer l'interface en quatre volets séparés :

- *Volet 1 : le(s) UKI affecté(s)*. Ce volet désigne les UKI (un ou plusieurs) affectés par le changement dont l'analyse a été visualisée auparavant (ici le changement `MergeClasses`).

- *Volet 2 : les classes appropriées à la modification des UKI affectés*. Ce volet présente les classes identifiées comme étant pertinentes à la modification des UKI présentés dans le volet 1. Les classes sont énumérées en fonction de leur degré de pertinence, en commençant par celles que le système considère comme étant les plus pertinentes et en terminant avec celles qui le sont moins.

- *Volet 3 : les caractéristiques des classes appropriées pour la modification des UKI*. Ce volet se compose des commentaires et des caractéristiques propres à chacune des classes listées dans le volet 2. La section *Comments* décrit pour quelle raison les classes ont été considérées comme étant appropriées à la modification des UKI. Les sections *SubClasses*, *DisjointClasses* et *EquivalentClasses*, indiquent si des (sous)classes et des axiomes ont été transférés, effacés ou ajoutés à la classe sélectionnée. De même, la section *Properties* indique si des propriétés ont été transférées, effacées ou ajoutées à la classe sélectionnée.

- *Volet 4 : les UKI modifiés*. Ce volet présente les UKI modifiés.

L'utilisateur choisit parmi les suggestions et valide la modification des UKI

Pour interagir avec SAM, l'utilisateur sélectionne un UKI à partir du volet correspondant aux UKI à modifier, par exemple celui référant à `Professeur`, une des classes fusionnées lors du changement `MergeClasses`.

Ensuite il explore les classes suggérées par SAM comme étant appropriées à la modification de l'UKI sélectionné. Pour chacune de ces classes, l'utilisateur peut consulter une description de leurs caractéristiques, montrant, entre autres, pourquoi ces classes ont été considérées pertinentes. Pour accéder à la description, l'utilisateur n'a qu'à sélectionner une

classe dans la liste, par exemple la classe `ConcepteurCours`, et les informations qui y sont reliées s'affichent dans le troisième volet. Ainsi, il peut lire le commentaire affirmant qu'il s'agit de la classe résultante de la fusion, donc elle est recommandée pour la modification des UKI. Il peut aussi observer que des sous-classes lui ont été transférées à partir de classes fusionnées (p.ex. `Designer_MISA`, `ModelisateurConnaissances`), de même que des propriétés (p.ex. `identifieActivitesApprentissage`, `identifieConnaissances`) et des axiomes (p.ex. la disjonction avec `Pedagogue`).

Finalement, l'utilisateur choisit une ou plusieurs classes, parmi celles qui lui sont présentées, pour modifier les UKI sélectionnés. Il choisit, par exemple, la classe `ConcepteurCours` pour modifier l'UKI qui référait au `Professeur` et la classe `DesignerIMS_LD` pour modifier l'UKI qui référait au `DesignerPedagogique`. Il valide ses choix en appuyant sur le bouton *ModifyUKI*.

Le système SAM modifie les UKI selon les choix de l'utilisateur

Après la validation de la part de l'utilisateur, le système SAM modifie les UKI, tel que montré dans le quatrième volet de l'interface. À la fin du processus de modification assistée, SAM génère un fichier des UKI modifiés comprenant aussi les résultats issus de la modification (semi)automatique.

5.3.4. **Générer le fichier des UKI modifiés**

Après que les utilisateurs aient validé la modification de tous les UKI, que ce soit selon le mode (semi)automatique ou assisté, la dernière étape de la modification du référencement sémantique est celle où le système SAM génère le fichier contenant les UKI modifiés. Un exemple de ce type de fichier est fourni dans la Figure V-25.

Figure V-25. Fichier des UKI modifiés selon le mode (semi)automatique et assisté

Le fichier ci-dessus illustre quatre des UKI modifiés : les deux premiers selon le mode (semi)automatique, où uniquement l'identifiant de version est modifié, et les deux derniers selon le mode assisté, où tant l'identifiant de la version que celui de la classe sont modifiés.

5.4 **Conclusion**

Dans ce chapitre, nous avons répondu au troisième objectif de cette thèse, à savoir : l'identification des effets des changements sur le référencement sémantique des ressources ainsi que la conception des solutions pour maintenir l'accès et l'interprétation de ressources référencées par des concepts provenant des ontologies évolutives. Nous avons regroupé le

tout dans le système conseiller *SemanticAnnotationModifier* (SAM) qui, avec la notion de référencement sémantique évolutif, ouvre un chemin tout à fait nouveau dans notre domaine de recherche.

Nous avons commencé le chapitre en introduisant la notion d'UKI (*Uniform Knowledge Identifier*) comme moyen de spécification formelle du référencement sémantique des ressources. Nous avons illustré ensuite de quelle manière SAM procède à l'analyse des effets des changements, selon que les utilisateurs rendent ou non disponible un fichier des UKI (analyse contextuelle ou générale). Nous avons également décrit les trois volets de l'analyse effectuée : l'accès aux ressources, l'interprétation des ressources, les relations logiques entre versions d'ontologie. Précisons que cette analyse prend en compte tant les changements élémentaires que les changements complexes.

Finalement, nous avons proposé deux modalités pour la modification du référencement, les deux étant supportées actuellement par le système SAM : la modification (semi)automatique des UKI et la modification assistée, cette dernière étant principalement associée aux UKI affectés par des changements menant à une perte d'accès aux ressources. En ayant accès à l'historique des changements apportés à V_N pour obtenir V_{N+1}, le système SAM est capable d'offrir une large palette des solutions possibles pour modifier les UKI et préserver ainsi l'accès aux ressources au moyen de la nouvelle version de l'ontologie.

Dans le chapitre suivant, nous allons évaluer, auprès d'utilisateurs humains, le cadre de référence que nous avons conçu pour la gestion des changements apportés aux versions d'ontologie, cadre composé de deux modules informatiques : le CHB et le SAM.

Chapitre VI

EVALUATION DES SYSTÈMES : *CHANGEHISTORY-BUILDER* ET *SEMANTICANNOTATION-MODIFIER*

Dans les chapitres précédents, nous avons discuté du cadre de référence que nous proposons pour la gestion des changements apportés aux versions d'ontologie. Ce cadre est composé de deux systèmes de base : le *ChangeHistoryBuilder* (CHB), orienté vers l'identification des changements, et le *SemanticAnnotationModifier* (SAM), orienté vers l'analyse et la résolution des effets des changements sur le référencement sémantique des ressources.

Dans ce chapitre, nous présentons l'évaluation de ce cadre. Nous commençons par expliquer le choix du critère d'utilité ainsi que les techniques utilisées pour l'évaluation du CHB et du SAM (*cf.* Section 6.1). Dans la Section 6.2, nous décrivons les démarches et les objectifs d'évaluation pour chacun de ces deux systèmes. Dans la Section 6.3, nous présentons l'analyse des données et les conclusions tirées concernant l'utilité du CHB et du SAM. Nous terminons avec une conclusion du chapitre (*cf.* Section 6.4).

6.1 **Caractéristiques de l'évaluation des systèmes**

6.1.1. **Type d'évaluation**

Dans un travail de recherche, on peut considérer une évaluation comme un processus systématique de collecte d'informations et d'analyse de ces informations, un processus qui aboutit à un jugement de valeur portant sur (Gediga, Hamborg, et Düntsch, 2002) : (1) la comparaison des logiciels différents ou des prototypes ou versions d'un même logiciel dans le but d'identifier celui qui correspond le mieux aux besoins exprimés ; (2) l'identification du degré de qualité d'un système selon divers critères d'analyse ; (3) l'analyse des points faibles d'un système dans le but d'identifier des développements futurs ou de réaliser la réingénierie du système.

L'évaluation des systèmes *ChangeHistoryBuilder* (CHB) et *SemanticAnnotationModifier* (SAM) se range dans la deuxième catégorie, celle ayant comme but général d'identifier le degré de qualité selon un ensemble de critères d'analyse. Les critères d'utilité et d'utilisabilité des systèmes sont parmi les plus prisés dans le domaine de l'informatique (Grudin, 1992; Nielsen, 1993), puisqu'ils mettent de l'avant deux dimensions de l'évaluation des systèmes qui, jusqu'à ce moment, étaient confondus. Ainsi, l'utilité et l'utilisabilité des systèmes apportent des réponses à la question : pourquoi des systèmes avec de faibles qualités ergonomiques sont très appréciés par leurs utilisateurs, tandis que d'autres, avec de fortes qualités ergonomiques n'apportent pas de services essentiels et donc sont peu utilisés ? Dans la section suivante, nous présentons ces deux critères et nous justifions notre choix du critère de l'utilité pour l'évaluation des systèmes CHB et SAM.

6.1.2. **Critères d'évaluation**

6.1.2.1. **Utilité et utilisabilité**

La norme ISO 9241-11 (1998) définit l'utilisabilité comme le degré selon lequel un système peut être utilisé, par des utilisateurs identifiés, pour atteindre les résultats prévus, avec un minimum d'effort et avec de la satisfaction, dans un contexte d'utilisation désigné.

On retrouve imbriqués dans cette définition, deux aspects primordiaux de l'évaluation des systèmes : (1) le système peut faire ce qui lui est exigé – la notion d'utilité ; (2) le système peut être facilement utilisé et ses fonctionnalités facilement apprises – la notion d'utilisabilité. À la suite de Grudin, (1992), Nielsen, (1993), distingue la notion d'utilité de celle d'utilisabilité :

- l'**utilité** (*utility*) dénote la conformité entre les buts et les résultats que le système permet d'atteindre, pour un domaine, une exploitation et un environnement donnés. Mesurer l'utilité revient alors à identifier le degré de pertinence du système, considéré dans un contexte spécifique, et à décider de son aspect fonctionnel, c'est-à-dire la présence de fonctionnalités qui répondent à ce qu'on attend du système.

- l'**utilisabilité** (*usability*) traite plus l'aspect opérationnel du système, c'est-à-dire la possibilité d'utiliser, d'une manière conviviale, ses fonctionnalités dans un contexte précis et désigne un ensemble de critères d'évaluation, comme la facilité d'apprentissage et d'appropriation du système, l'efficience (le fait d'atteindre, sans perdre trop de temps, le but que l'on s'est fixé) ou la fiabilité (la prévention ou la gestion des erreurs par le système).

Deux aspects importants retiennent notre attention. Premièrement, on peut observer que, selon Nielsen (1993), le critère d'efficience, c'est-à-dire un critère lié au but, et pas seulement à l'utilisation, est considéré comme un critère d'utilisabilité. Certains spécialistes, comme Meads et Stubbs, (2001), vont même jusqu'à intégrer le critère d'utilité parmi les critères d'utilisabilité, ce qui revient à interpréter tout problème d'un système comme lié à l'utilisabilité de celui-ci. Tricot, (2001), clarifie cette confusion, en démontrant qu'elle est produite par les liens existants entre la notion d'utilité et celle d'utilisabilité, considérées comme des notions bien distinctes l'une de l'autre. Ainsi, selon Tricot, un système peut respecter tous les critères d'utilisabilité mais être inutile puisqu'il ne répond pas aux attentes des utilisateurs. De même, un système utile peut être difficilement utilisable si, par exemple, il demande un effort cognitif intense pour apprendre et mémoriser 'comment ça marche' ou 'ce qu'il fait'. Ainsi, pour qu'un système soit convenable, il doit être à la fois utile et utilisable.

Le deuxième aspect réfère au contexte de l'utilisation du système. L'utilité et l'utilisabilité ne sont, ni l'une ni l'autre, des qualités intrinsèques d'un système, mais sont relatives au contexte dans lequel il est utilisé et au profil des utilisateurs. Chaque évaluation d'un système doit donc respecter certaines restrictions relatives au contexte de l'usage (Gediga, Hamborg, et Düntsch, 2002) :

- les caractéristiques des utilisateurs, comme l'expérience ou les intérêts ;
- l'environnement de l'évaluation, allant de conditions de laboratoire contrôlées à de conditions non-structurées/contrôlées ;
- la nature de l'objet évalué, qui peut être un prototype papier, un prototype partiellement fonctionnel ou un système de type exploratoire ou encore un système complètement finalisé.

6.1.2.2. **Choix du critère d'utilité**

Les systèmes CHB et SAM sont les premiers prototypes dédiés à la fois à la gestion des changements apportés aux ontologies distribuées et à la résolution des effets des changements sur le référencement sémantique des ressources. Ces prototypes, vus comme des systèmes '**exploratoires des idées**' plutôt que des systèmes finalisés, nécessitent donc une évaluation dont l'accent est mis sur la validation de la conceptualisation et des fonctionnalités envisagées et non sur la facilité d'utilisation ou sur l'ergonomie des interfaces.

Par conséquent, le but principal est de valider les fonctionnalités du CHB et du SAM en **termes d'utilité** plutôt qu'en termes d'utilisabilité. On peut cependant se poser la question s'il est convenable d'évaluer l'utilité d'un système sans tenir compte de son utilisabilité ? Si la conception de systèmes est fondée sur une forte analyse des besoins afin de bien identifier les exigences de futurs utilisateurs (comme dans le monde de l'industrie, par exemple), l'utilité du système est déjà prévalidée dans une grande mesure et c'est l'utilisabilité qui doit prévaloir durant la conception et lors de l'évaluation des systèmes (Löwgren, 1995). Par contre, si on se situe dans un contexte de recherche, un contexte où ce qui prévaut est l'innovation, l'apport des idées novatrices qui peuvent, par elles-mêmes, faire émerger de nouveaux besoins, ce qu'on doit valider d'abord c'est l'utilité, l'intérêt que suscitent les

systèmes implémentant ces idées, les buts que ces systèmes permettent d'atteindre ainsi que le degré d'atteinte des buts. Ceci est encore plus vrai si ce qu'on veut évaluer sont des prototypes exploratoires utilisés dans de nouveaux contextes, comme celui du Web sémantique éducatif.

6.1.3. **Approche et techniques d'évaluation**

6.1.3.1. **Approche qualitative pour l'évaluation des systèmes**

L'approche qualitative vise à étudier en profondeur un objet ou un phénomène. Elle consiste à collecter des données qualitatives (paroles, comportements, textes) sur l'objet ou le phénomène et à analyser ces données. La principale différence avec l'approche quantitative, qui consiste à colliger des données numériques relatives à un certain nombre de variables, se situe au niveau de l'analyse et des conclusions tirées. L'analyse quantitative, dont l'analyse statistique est l'exemple typique, est une démarche de quantification des données, provenant d'un grand nombre de participants, dans le but d'obtenir des résultats reproductibles et généralisables et d'exclure les particularités individuelles et subjectives. Quant à l'analyse qualitative, elle est plutôt une démarche de recherche du sens (Paillé et Mucchielli, 2003), une démarche pendant laquelle le chercheur observe des comportements ou interprète des paroles ou des textes, afin de caractériser un objet ou un phénomène et d'en trouver ainsi son sens, sa pertinence même. Ce sens dégagé peut servir aux fins de la découverte (l'analyste cherche à trouver des éléments et à les agencer pour comprendre l'objet ou le phénomène à l'étude) ou pour la vérification (en partant de certains constats, l'analyste recueille des données qui servent à appuyer ses hypothèses, à vérifier des aspects ou même à qualifier des résultats quantitatifs). Ainsi, bien que les résultats qualitatifs ne soient pas nécessairement généralisables, ils sont exploratoires et explicatifs et servent souvent comme point de départ dans la validation de nouveaux concepts ou dans l'ingénierie d'un produit.

6.1.3.2. **Techniques d'évaluation qualitative utilisées**

Dans cette section, nous donnons un bref aperçu des techniques que nous avons utilisées lors de l'évaluation qualitative du système CHB et du système SAM. Elles sont au nombre de

cinq : la verbalisation, l'évaluation par paire, les questionnaires, l'entretien et le focus-groupe.

Afin de pouvoir capter l'activité cognitive des utilisateurs en cours de travail, la technique de **verbalisation** (*thinking aloud*) demande à chaque participant de réfléchir à haute voix, à exprimer ses raisonnements lors de l'interaction avec le système à évaluer (Jorgensen, 1989). Elle demande : (1) aux participants d'exprimer verbalement ce qu'ils pensent, ce qu'ils ressentent par rapport aux fonctionnalités d'un système et (2) aux évaluateurs d'enregistrer les données ainsi exprimées. La verbalisation peut s'effectuer pendant l'interaction avec le système – verbalisation concurrente - ou après l'interaction – verbalisation rétrospective. Ericsson et Simon, (1980), démontrent que les données issues d'une verbalisation rétrospective sont plus riches en explications, alors que celles produites par une verbalisation simultanée sont plus pertinentes puisqu'elles relèvent de l'expérience immédiate des participants.

L'**évaluation par paire** est une technique où les participants sont regroupés en groupe de deux pour réaliser les activités d'évaluation. Les avantages de cette technique sont multiples (Gediga, Hamborg, et Düntsch, 2002; O'Malley, Draper, et Riley, 1984). Elle permet la création d'une interaction constructive, elle demande aux participants d'argumenter leurs idées et leurs opinions (et non pas simplement de les énoncer, comme dans la verbalisation individuelle) pour en arriver à une solution collective, elle supporte la motivation dans le travail d'évaluation ainsi que la créativité et la découverte de nouveaux aspects et fonctionnalités. C'est pourquoi Gediga, Hamborg et Düntsch, (2002), la recommandent principalement pour l'évaluation de prototypes exploratoires.

Les **questionnaires** (Quivy et vanCampenhoud, 1995) sont des listes de questions écrites, adressées aux participants lors d'une évaluation. On peut utiliser un questionnaire à deux fins : soit pour obtenir une validation statistique ou qualitative d'hypothèses émises, soit pour rassembler des opinions ou des suggestions quant au degré d'utilité et d'utilisabilité des systèmes. Il existe trois principaux types de questions : (1) les questions fermées avec une série de réponses à choisir ; (2) les questions ouvertes qui ne prévoient aucune réponse, mais demandent aux participants de s'exprimer librement ; (3) les questions semi-ouvertes qui sont une combinaison des deux autres.

L'**entretien** est une technique d'évaluation où on pose des questions aux participants (experts ou non) afin de rassembler de l'information sur un sujet particulier (Quivy et vanCampenhoud, 1995). En fonction du degré de structuration de l'entretien, on distingue trois grands types d'interviews : (a) l'entretien structuré, qui consiste à interroger plusieurs sujets à l'aide du même protocole et de la même liste de questions, déterminée à l'avance; (2) l'entretien semi-structuré, qui se caractérise par la combinaison des questions fondamentales et d'une série de questions secondaires à poser ou non en fonction de l'évolution de l'entretien; (3) l'entretien non-structuré ou libre, qui consiste à inciter le répondant à s'exprimer en long et en large afin de saisir ses motivations profondes.

La technique du **focus-groupe** désigne une discussion de groupe, structurée par un scénario, défini d'avance, qui charpente le déroulement de la discussion (Slocum, 2006). Un focus groupe rassemble plusieurs personnes invitées à faire des réflexions sur un sujet précis et à réagir par rapport aux opinions et aux affirmations des autres. Cette technique donne ainsi la possibilité de dégager une grande multiplicité de points de vue, de faire le point des connaissances, opinions, souhaits et pratiques de différents types d'acteur, de déterminer le degré de consensus existant sur un sujet donné.

6.2 **Démarche d'évaluation des systèmes CHB et SAM**

Dans cette section, nous présentons les participants, l'environnement ainsi que la procédure que nous avons mis en œuvre pour l'évaluation des systèmes CHB et SAM.

6.2.1. **Contexte d'évaluation**

6.2.1.1. **Participants à l'évaluation**

Les critères établis pour recruter des sujets représentatifs étaient : une connaissance des ontologies formelles et d'un éditeur d'ontologies (particulièrement MOT+Onto[51]), une

[51] MOT+Onto est un éditeur graphique des ontologies OWL, développé au centre de recherche LICEF. Il est accessible à l'adresse `http://www.licef.teluq.uquebec.ca/fr/index.htm`.

connaissance du langage OWL de représentation des ontologies et finalement une expérience dans la conception pédagogique. Nous avons choisi ce dernier critère afin d'identifier l'utilisation potentielle des systèmes CHB et SAM dans le domaine de la conception pédagogique des systèmes d'apprentissage fondés sur le référencement des ressources.

La participation s'est faite sur une base volontaire après l'envoi d'un courrier électronique expliquant les objectifs de l'expérimentation, les modalités de participation et le temps requis. Six personnes[52] ont été retenues (*cf.* Tableau VI-1).

Tableau VI-1. Participants sélectionnés pour l'évaluation des systèmes

Participant	Niveau d'expérience			
	Conception Pédagogique	**Ontologies formelles**	**Éditeur ontologie**	**Langage OWL**
P1	Bon	Très bon	Très bon (MOT+Onto)	Très bon
P2	Bon	Très bon	Bon (MOT+Onto)	Bon
P3	Faible	Bon	Bon (MOT+Onto)	Bon
P4	Bon	Bon	Bon (MOT+Onto)	Moyen
P5	Très bon	Bon	Très bon (MOT+Onto)	Moyen
P6	Très bon	Moyen	Moyen (MOT+Onto)	Faible

6.2.1.2. **Environnement d'évaluation**

L'expérimentation a été réalisée le 15 et le 20 mai 2007 au Laboratoire-Observatoire de Recherche en Ingénierie du Téléapprentissage (LORIT) du centre de recherche LICEF. Le LORIT a une vocation purement scientifique, ce laboratoire s'adressant en priorité aux chercheurs pour leurs expérimentations et la diffusion des résultats de recherche.

Concrètement parlant, le LORIT permet l'observation et la cueillette de données multimédias de multiples sources, avec synchronisation de marquage temporel. Il est constitué de deux locaux : une salle d'expérimentation et une salle d'observation (*cf.* Figure

[52] Toute personne participante a signé une entente d'agrément issue d'un dossier de déontologie sur l'éthique de recherche.

VI-1). La **salle d'expérimentation** est un vaste local muni de onze postes de travail et de l'équipement de diffusion multimédia standard (écran, rétroprojecteur, etc.). Elle est par ailleurs munie d'équipements d'enregistrement vidéo et audio, à la fois au niveau de la salle (micro au plafond et trois caméras mobiles) et au niveau de chacun des postes de travail (webcam et micro de proximité). Un contrôleur (écran tactile) permet de gérer l'ensemble des équipements depuis la salle d'expérimentation ou de la salle d'observation.

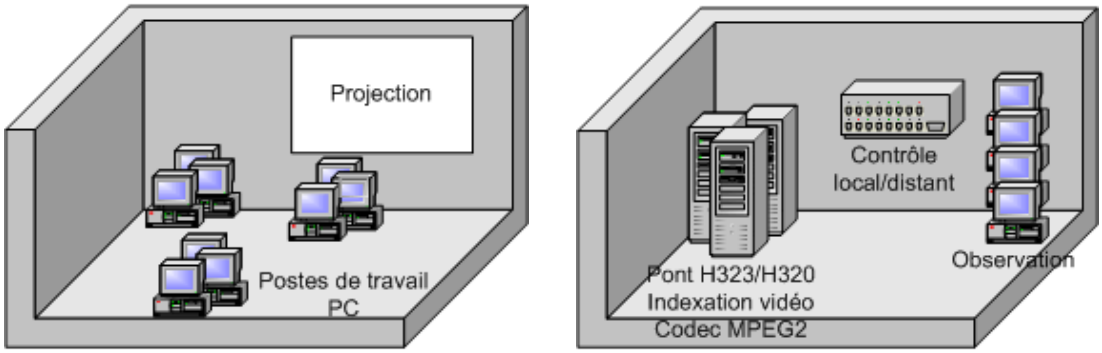


Figure VI-1 (a) la salle d'expérimentation ; (b) la salle d'observation

La **salle d'observation** permet, via un dispositif technique de haute performance, l'observation directe de ce qui se passe dans la salle d'expérimentation grâce à un ensemble d'écrans qui fournissent les images captées par les différentes caméras vidéo et le reflet de chacun des onze postes de travail. Elle permet aussi, l'enregistrement des images pour chaque caméra et chaque ordinateur, ainsi que la capture audio du micro du plafond et de ceux de proximité (reliés individuellement aux ordinateurs).

6.2.2. **Procédure d'évaluation**

Les étapes de l'expérimentation sont au nombre de cinq : (1) la présentation de la recherche ; (2) l'évaluation de la visualisation des changements avec CHB ; (3) l'évaluation de l'analyse des effets des changements fournie par SAM ; (4) le focus-groupe et la mise en contexte de la modification du référencement avec SAM ; (5) l'inspection guidée de la modification du référencement avec SAM.

6.2.2.1. **Étape 1 - présentation de la recherche**

La première étape a comme objectif de permettre à tous les participants de s'approprier le concept de l'évolution des ontologies. Elle vise également de les rendre conscients du contexte, des objectifs et de la nature concrète de notre recherche et ainsi, de leur fournir les connaissances nécessaires pour porter des jugements concernant l'utilité des services proposés par les systèmes CHB et SAM.

6.2.2.1.1. *Objectifs visés par l'étape*

Puisqu'il s'agit d'une étape préliminaire à l'expérimentation, nous n'avons prévu aucun objectif concret d'évaluation, mais seulement une familiarisation des participants avec notre recherche ainsi qu'avec un certain nombre des notions utilisées lors de l'expérimentation. Ceci dans le but de permettre aux participants de s'approprier les termes spécifiques qu'ils rencontreront durant l'expérimentation et, de cette manière, de leur permettre de se concentrer davantage sur l'évaluation de l'utilité des systèmes.

6.2.2.1.2. *Déroulement de l'étape*

Participants

Pour s'assurer que les sujets possèdent les connaissances théoriques de base pour effectuer les tâches prévues, ils ont tous été convoqués de participer à la présentation de notre recherche.

Procédure

Après avoir discuté la problématique de notre recherche (le contexte, les objectifs et les résultats), nous avons présenté ce qu'est: (1) une ontologie OWL ; (2) l'éditeur MOT+Onto que nous avons utilisé pour permettre l'exploration des versions d'ontologie. Nous avons illustré les notions que les participants rencontreront lors des activités d'évaluation, par exemple la notion de changement élémentaire et complexe, primaire et additionnel, la notion du référencement sémantique et celle de l'accès direct et indirect aux ressources référencées. Nous avons conclu cette étape en attirant l'attention sur le fait que notre but principal est

celui d'évaluer l'utilité des systèmes CHB et SMA et en donnant des indications sur le protocole à suivre lors de l'expérimentation.

Instruments

Pour exposer la recherche d'une manière conviviale, nous avons utilisé le dispositif de projection sur grand écran mis à notre disposition dans le LORIT ainsi qu'une présentation *PowerPoint* que nous avons conçu pour ces fins. Des documents explicatifs ont été remis aux participants.

6.2.2.1.3. *Collecte des données*

Cette étape ne nécessite aucune collecte des données.

6.2.2.2. **Étape 2 – évaluation du système CHB**

Nous avons d'abord conçu le CHB comme un système de traçage et de description formelle des changements. Plus tard nous nous somme rendue compte que, bien que le CHB a une forte composante formelle, il peut également être très utile pour offrir une vue sur l'évolution des ontologies à des utilisateurs humains. Cette étape vise l'évaluation de l'utilité de la fonctionnalité du CHB, permettant l'identification et la visualisation des changements apportés à une version V_N pour obtenir une nouvelle version V_{N+1}.

6.2.2.2.1. *Objectifs visés par l'étape*

Le terme *utilité* dénote la pertinence fonctionnelle d'un système : le système possède-t-il des fonctionnalités qui répondent à nos besoins et qui nous permettent d'atteindre les objectifs fixés ou le niveau de compréhension désiré ? Dans le contexte novateur de l'évolution des ontologies, on ne peut cependant pas encore parler de besoins (ou objectifs) bien spécifiés, mais plutôt émergeant au fur et à la mesure que les utilisateurs avancent dans la compréhension du domaine. Ainsi, la pertinence fonctionnelle du CHB (également valable pour SAM) se traduit par les questions suivantes :

- le système offre-t-il de nouvelles fonctionnalités qu'on trouve appropriées dans le contexte d'évolution des ontologies ?

- nous permet-il d'acquérir une nouvelle compréhension du processus d'évolution ou d'enrichir celle que nous possédons déjà ?

- nous ouvre-t-il de nouvelles possibilités de comprendre et de gérer les versions d'ontologie ?

Dans cette étape, les objectifs de l'évaluation sont axés sur la fonctionnalité d'identification des changements et visent à identifier comment cette fonctionnalité influence les utilisateurs dans la compréhension du processus d'évolution des ontologies. Ces objectifs permettent de déterminer les points forts de CHB ainsi que les améliorations indispensables pour son effectivité. Ils sont au nombre de trois :

- **Objectif 1** (étape 2). Le premier objectif est de déterminer le degré d'influence du CHB sur la compréhension du processus d'évolution d'une ontologie. Par CHB, nous comprenons ici la fonctionnalité permettant l'identification des changements et des stratégies[53] d'évolution appliquées.

- **Objectif 2** (étape 2). Les changements ontologiques peuvent être élémentaires ou complexes. Le deuxième objectif est de déterminer dans quelle mesure le fait de visualiser des changements complexes, en plus des changements élémentaires, permet une meilleure compréhension de la logique d'évolution ainsi que de 'l'intention formelle' de l'acteur qui a modifié l'ontologie (c.-à-d. l'ontologiste). Le terme 'formel' veut mettre en évidence que nous ne parlons pas ici de ce que l'ontologiste avait comme motivation, par exemple modifier la description de la classe `DesignerPédagogique` pour répondre aux modifications apparues dans le domaine du téléapprentissage. Nous parlons plutôt des types de changements qu'il voulait apporter, par exemple la fusion des classes `DesignerPédagogique` et `Professeur` en `ConcepteurCours`, non pas une suite d'effacements et d'ajouts de

[53] Les stratégies d'évolution réfèrent aux relations de causalité entre les changements primaires et additionnels.

classes, qui, même si elle produit le même résultat en V_{N+1}, ne possède pas la même sémantique formelle.

- **Objectif 3** (étape 2). Un système adéquat est un système qui est, en même temps, utile et utilisable. Le troisième objectif poursuivi dans cette étape est celui d'identifier les principaux problèmes d'utilisabilité et de proposer des pistes de solutions.

6.2.2.2.2. *Déroulement de l'étape*

Techniques d'évaluation

Nous avons utilisé trois techniques d'évaluation imbriquées : la technique du questionnaire pour permettre aux participants de donner leur avis, mais aussi pour les guider dans les activités d'évaluation, la verbalisation pour extérioriser les processus mentaux qui surviennent lors de ces activités et finalement, l'évaluation par paire pour motiver les participants à argumenter leurs opinions.

Participants

Les six participants ont été organisés en groupe de deux (*cf.* Tableau VI-2).

Tableau VI-2. Organisation des participants par dyade

Numéro dyade	Participants	Caractérisation de la dyade
Dyade no. **1**	P2 et P6	Cette dyade est caractérisée par un fort axe pédagogique. En même temps, un des partenaires est capable de faire comprendre à l'autre (au besoin) les notions sous-jacentes aux ontologies OWL.
Dyade no. **2**	P1 et P3	Cette dyade est caractérisée par une forte connaissance des ontologies OWL. Un des partenaires est assez bon en conception pédagogique, ce qui fait qu'il peut apporter dans la dyade une perspective pédagogique.
Dyade no. **3**	P4 et P5	Cette dyade est caractérisée par un équilibre entre les deux participants, tant en ce qui concerne les ontologies OWL que la conception pédagogique.

En organisant les dyades d'une telle manière, notre intérêt était double : (1) de combler les carences des participants (que ce soit en ontologies OWL ou en conception pédagogique)

en les associant avec un partenaire plus compétent dans le domaine plus faible et (2) de créer de dyades plus spécialisées dans un domaine ou dans un autre pour permettre la comparaison des résultats issus de l'analyse des données qualitatives.

Procédure

Cette étape s'est déroulée de la manière suivante. Premièrement, chaque dyade était tenue de répondre à un questionnaire ouvert – noté *PréTest_A.1* – qui avait pour but d'apprécier l'importance donnée par les participants au fait de connaître si et comment une ontologie a évolué. Le temps prévu pour cette activité était de 5 à maximum 10 minutes.

Nous avons ensuite demandé aux dyades d'explorer, à l'aide du l'éditeur MOT+Onto, deux versions (*version_2* et *version_3*) d'une ontologie nommée *OntologieActeursTeleApprentissage*. Après un temps d'exploration d'environ 10 minutes, les dyades devaient, à l'aide du *Test_B.1*, indiquer si l'ontologie a évolué d'une manière significative ou non, justifier leur réponse et préciser les changements appliqués. Ceci pour permettre une comparaison entre la compréhension du processus de l'évolution, développée par les sujets sans et avec l'aide du CHB, et de répondre ainsi à l'objectif 2 (étape 2).

Dans un troisième temps, les participants ont exploré les changements à l'aide du système CHB. Pour leur éviter des manipulations n'ayant aucun lien avec nos objectifs d'évaluation, comme le téléchargement des versions d'ontologie par exemple, les participants ont pu accéder directement à la fenêtre de visualisation des changements apportés à la deuxième version de l'ontologie *OntologieActeursTA* pour obtenir la troisième. Cette fenêtre – nommée *VisualisationChangements_1* – présente (a) un ensemble des changements élémentaires et (b) un ensemble des changements complexes. Les participants ont répondu ensuite à un questionnaire semi-ouvert – noté *Test_B.2* – conçu pour leur permettre d'exprimer ce qu'ils ont compris après l'exploration des changements avec CHB. En analysant ensuite la verbalisation produite lors de l'exploration avec et sans CHB ainsi que les deux questionnaires (Test_B.1 et Test_B.2) il nous est donc possible de déterminer si le système aide à la compréhension du processus d'évolution (objectif 1 de l'étape 2). Le temps prévu pour cette activité était d'environ 25 minutes.

Finalement, les participants ont consulté une deuxième fenêtre de visualisation des changements – nommée *VisualisationChangements_2* – dans laquelle des changements élémentaires sont présentés sous une forme complexe,. Par exemple, l'effacement des classes `DesignerPedagogique` et `Professeur` est remplacé par la fusion de ces deux classes. Ensuite, un questionnaire ouvert – le *Test_B.3* – a été fourni aux participants pour demander leur avis sur l'utilisation des changements complexes : produisent-ils une 'meilleure' compréhension de l'évolution, la 'vraie' logique de l'évolution est-elle plus facile à saisir, les changements complexes sont-ils plus 'utilisables'? Le but ici est de déterminer si, pour les mêmes versions V_N et V_{N+1} le fait d'avoir accès aux changements complexes, et non uniquement à une suite des changements élémentaires, a des bénéfices cognitifs (objectif 2 de l'étape 2). Le temps prévu pour cette activité était d'environ 10 minutes.

Pendant le déroulement de cette étape, tous les participants étaient tenus de discuter avec leur partenaire et de réfléchir à voix haute.

Instrumentation

Lors de cette étape nous avons utilisé plusieurs instruments pour préciser et supporter l'expérimentation : l'éditeur MOT+Onto et les deux versions de l'ontologie *OntologieActeursTA*, les quatre questionnaires ouverts et semi-ouverts (les PréTest_A.1, Test_B.1, Test_B.2, Test_B.3), et finalement, les deux fenêtres de visualisation des changements, fournies par le système CHB. Certains instruments peuvent être consultés dans l'Appendice E.

a. L'éditeur MOT+Onto et les versions d'ontologie

Nous avons choisi d'utiliser l'éditeur d'ontologies en mode graphique MOT+Onto puisqu'il est connu et même utilisé par la totalité des participants.

Pour obtenir les versions d'ontologie nous avons transposé sous une forme ontologique, à l'aide de l'éditeur MOT+Onto, la description des rôles des acteurs du téléapprentissage (Paquette, 2002), en introduisant cependant un ensemble de modifications menant à l'obtention de deux versions de l'ontologie. Ces deux versions sont présentées dans la Figure VI-2 et dans la Figure VI-3.

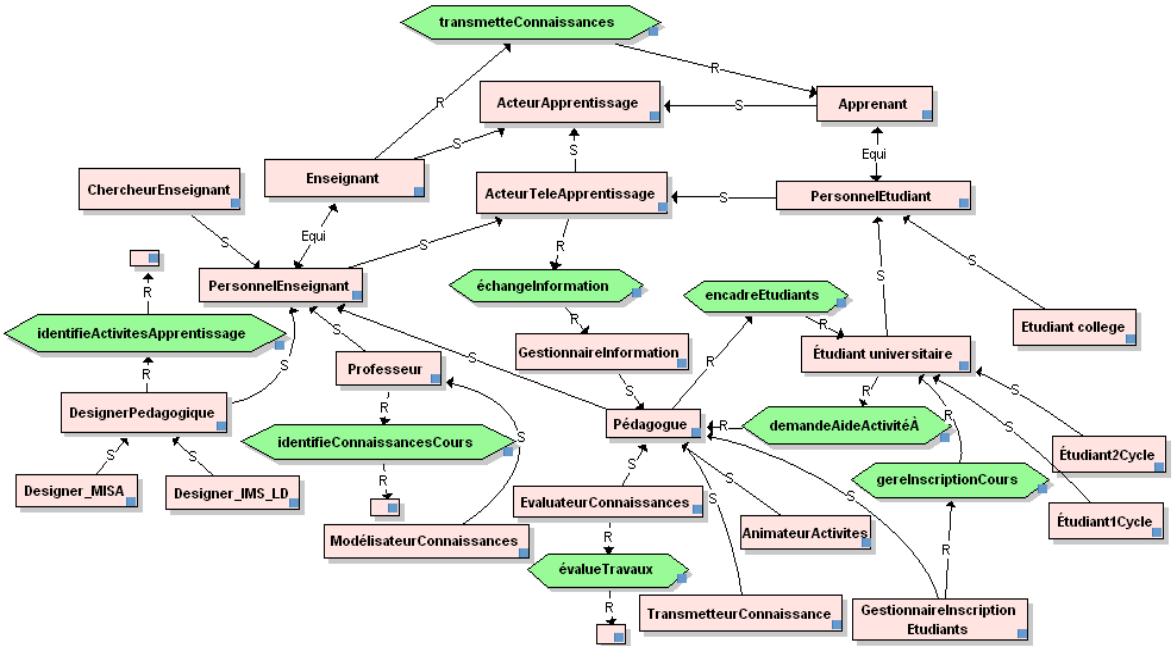


Figure VI-2. OntologieActeursTeleApprentissage_version_2

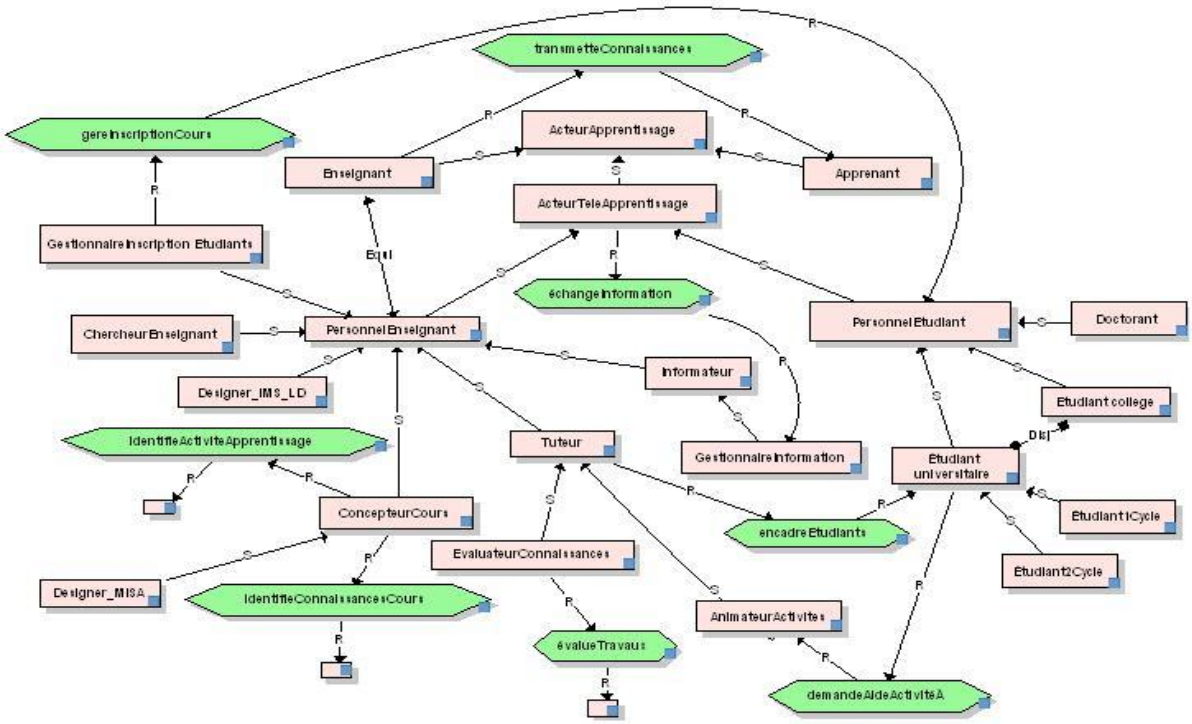


Figure VI-3. OntologieActeursTeleApprentissage_version_3

b. Les fenêtres de visualisation des changements

Le système CHB est tout d'abord un système conçu pour la journalisation de changements et pour une communication machine,. Son interface est donc une simple fenêtre de visualisation, peu conviviale pour le moment. Les deux fenêtres de visualisation utilisées sont présentées dans la Figure VI-4 et dans la Figure VI-5.

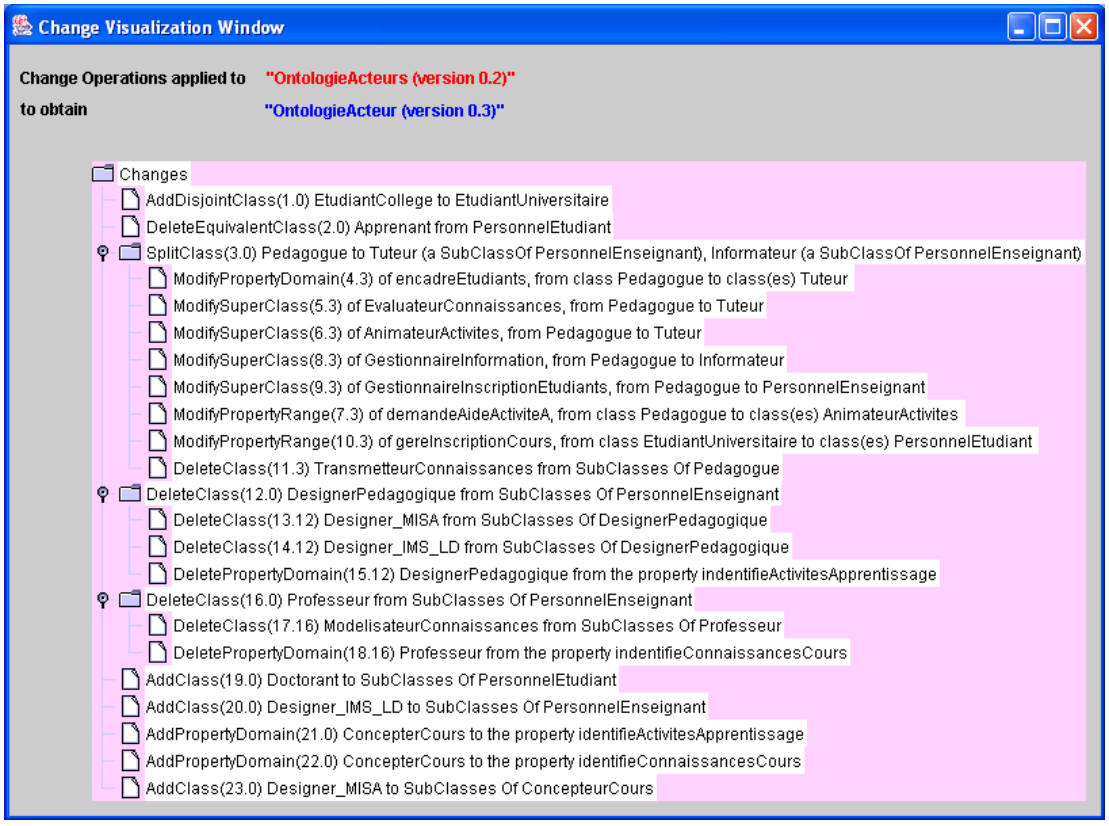


Figure VI-4. Fenêtre VisualisationChangements_1

La fenêtre VisualisationChangements_2 présente une vue différente de l'évolution pour la même version d'ontologie, une vue où le vrai changement de fusion entre `DesignerPedagogique` et `Professeur` est indiqué, non pas une suite des changements élémentaires, comme l'indique la première fenêtre.

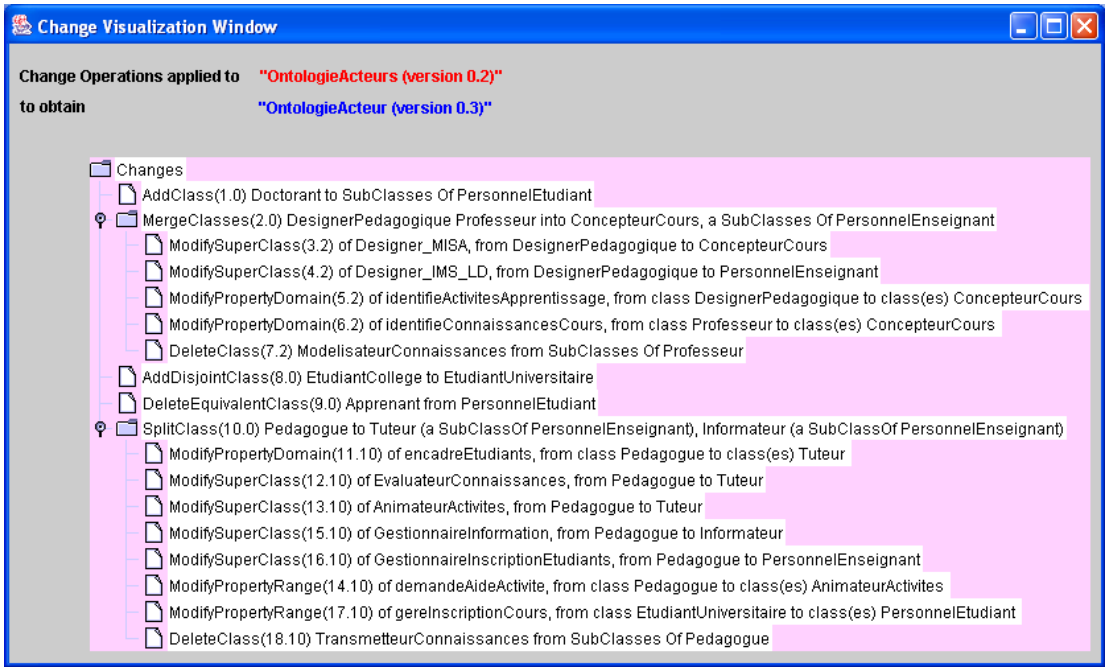


Figure VI-5. Fenêtre VisualisationChangements_2

c. Les questionnaires

Le **PréTest_A.1** se compose de deux questions demandant aux participants s'ils jugent nécessaire de savoir si et comment une ontologie a été modifiée. Chacune de ces questions devait être regardée d'abord du point de vue d'un acteur explorateur et ensuite d'un acteur client de l'ontologie. Un *acteur explorateur* est toute personne qui consulte l'ontologie dans le but de s'informer sur la conceptualisation d'un domaine. Un *acteur client* est toute personne qui utilise l'ontologie à des fins spécifiques (p.ex. le référencement sémantique des ressources ou la conception d'un cours dont la structure des connaissances est fondée sur l'ontologie).

Le **Test_B.1** demande aux participants de visualiser uniquement les deux versions d'ontologie et de préciser ensuite si l'ontologie a été modifiée d'une manière significative ou non, tout en justifiant leur réponse.

Le **Test_B.2** se compose de quatre questions conçues pour évaluer la compréhension développée par les participants lors de l'exploration des changements avec CHB, en utilisant

la première fenêtre de visualisation des changements. Premièrement, la même question que celle du Test_B.1 a été posée pour donner aux participants la possibilité de modifier leur opinion, à la suite de l'utilisation du CHB. Les deux autres questions demandent de décrire librement les types de changements apportés, en pointant cependant sur ceux plus difficiles à saisir, comme la division de la classe `Pédagogue`, ainsi que le contexte d'apparition des changements additionnels comme le transfert d'`AnimateurActivites` à la classe `Tuteur`. Enfin, la dernière question invite les participants à préciser l'intention formelle à la base des changements concernant la classe `DesignerPédagogique`.

Le **Test_B.3** a pour objet la deuxième fenêtre de visualisation des changements. Il questionne les participants sur la *nouvelle* intention formelle sous-jacente aux *nouveaux* changements relatifs à la classe `DesignerPedagogique`, c'est-à-dire la fusion et ses changements additionnels. Il demande aussi quelle fenêtre leur semble plus significative en ce qui concerne la compréhension de la *vraie* logique de l'évolution.

6.2.2.2.3. *Collecte des données*

La récupération des données issues de la verbalisation des participants a été réalisée en enregistrant (image et son) chaque dyade séparément à l'aide des micros de proximité et des caméras attachées à chaque ordinateur. Ainsi, à la fin de l'étape 2, nous avons disposé des données suivantes : les verbalisations des participants, les questionnaires dûment complétés, des notes prises durant l'expérimentation.

6.2.2.3. **Étape 3 – évaluation du système SAM (analyse des effets des changements)**

6.2.2.3.1. *Objectifs visés par l'étape*

Cette étape concerne l'évaluation de l'analyse des effets des changements fournie par le système SAM. Les objectifs d'évaluation sont au nombre de deux :

- **Objectif 1** (étape 3). Le premier objectif est de déterminer si la visualisation des effets des changements est pertinente pour un acteur 'client' de l'ontologie.

- **Objectif 2** (étape 3). Le deuxième objectif est celui de déterminer dans quelle mesure l'analyse présentée par SAM influence la compréhension des acteurs au sujet des effets des changements.

6.2.2.3.2. *Déroulement de l'étape*

Techniques d'évaluation

Nous avons utilisé les mêmes techniques et de la même manière qu'à l'étape 2.

Participants

Nous avons préservé la même organisation des participants qu'à l'étape 2.

Procédure

Premièrement, chaque dyade était tenue de répondre à un questionnaire ouvert – noté *PréTest_A.2* – portant sur l'importance donnée par les participants au fait de connaître les effets des changements apportés aux versions d'ontologies.

Deuxièmement, il a été demandé à chaque dyade d'explorer l'analyse des effets de changements[54]. Après un temps d'exploration de 10 minutes, les dyades ont complété un questionnaire semi-ouvert – noté *Test_C* – conçu pour leur donner la possibilité d'exprimer leur compréhension des effets des changements et leur avis sur la qualité sémantique de l'analyse pour un acteur client de l'ontologie.

Tout au long des activités d'exploration et de réponse aux questions, chaque participant était tenu de discuter avec son partenaire et de justifier, à voix haute, sa pensée. Le temps total prévu pour cette étape était de 15 à 20 minutes.

Instrumentation

Lors de cette étape, nous avons utilisé plusieurs techniques et outils comme l'analyse des changements, fournie par le système SAM, et deux questionnaires (PréTest_A.2 et Test_C). Ces techniques et outils peuvent être consultés dans l'Appendice E.

[54] Il s'agit de l'analyse des changements présentée dans la fenêtre VisualisationChangements_02.

a. L'analyse des changements fournie par SAM

Pour chaque changement, l'analyse est disponible dans une fenêtre accessible à partir de la fenêtre de visualisation des changements. Rappelons brièvement (*cf.* Figure V-13) que cette analyse se compose de trois volets : les deux premiers présentent l'effet du changement sur l'accès direct et indirect aux ressources et le troisième volet désigne les relations logiques entre versions d'ontologie.

b. Les questionnaires

Le **PréTest_A.2** demande aux participants s'ils considèrent comme nécessaire de connaître les effets des changements sur le référencement sémantique de ressources.

Le **Test_C** se compose des trois questions à choix multiples qui demandent aux dyades de choisir les affirmations vraies relatives à l'accès direct/indirect à une ressource R, annotée par une classe fusionnée (1re question), divisée (2e question) et par une classe d'où un axiome a été effacé (3e question). Ce test exige aussi l'opinion des participants au sujet de la validité sémantique de l'analyse et de l'intérêt qu'elle pourrait avoir pour la gestion du référencement sémantique.

6.2.2.3.3. *Collecte des données*

La récupération des données s'est déroulée de la même manière qu'à l'étape précédente.

6.2.2.4. **Étape 4 – focus-groupe et mise en contexte du SAM (modification du référencement)**

6.2.2.4.1. *Objectifs visés par l'étape*

L'objectif du focus-groupe était de discuter, en groupe, les points positifs et négatifs du CHB et SAM (analyse des changements) mais aussi de mettre en contexte la modification du référencement avec SAM, qui sera évaluée prochainement.

6.2.2.4.2. *Déroulement de l'étape*

Techniques d'évaluation

Pour cette étape, nous avons utilisé la technique du focus-groupe qui nous a permis d'appréhender une diversité de points de vue ainsi que les besoins ou les attentes de participants.

Participants

Tous les sujets ont participé au focus-groupe.

Procédure

Nous avons tenu le focus-groupe pendant une durée de 25 minutes, après les étapes 2 et 3. Pour mener la session, nous avons préalablement préparé un scénario et nous l'avons transposé sous la forme d'une série des questions à aborder. Pour chaque question du scénario, nous avons animé une discussion afin d'approfondir certains points, en centrant le débat si nécessaire.

Instruments

Pour le focus-groupe, nous avons utilisé une présentation *PowerPoint* avec les questions à débattre par les participants :

- **Question 1**. Voyez-vous les éditeurs d'ontologies comme des outils multifonctionnels permettant l'édition des changements élémentaires et complexes, mais aussi la gestion de leur historique?

- **Question 2**. En ce qui concerne la visualisation des changements, quelle configuration vous semble plus parlante, plus facile à comprendre et à manipuler, une visualisation à part ou une visualisation qui soit intégrée dans l'ontologie (*cf.* Figure VI-6) ? Dans le cas de cette dernière, comment la pensez-vous, sous une forme graphique ou sous la forme des commentaires textuels ajoutés à l'ontologie ?

- **Question 3.** Dans le contexte de la gestion des ressources référencées sémantiquement, que pensez-vous de l'effet des changements plus problématiques, tel que l'effacement ou la fusion des classes ? Considérez-vous nécessaire de

connaître les effets des changements sur le référencement ? Que pensez-vous de l'utilité d'un système qui vous fournirait ces informations et qui vous conseillerait aussi sur les solutions à prendre pour résoudre les effets problématiques ?

6.2.2.4.3. *Collecte des données*

Le focus-groupe a été enregistré (image et son) en utilisant le micro au plafond et les trois caméras mobiles disponibles dans la salle d'expérimentation du LORIT. Les fichiers obtenus ont été gravés sur un cédérom pour assurer un stockage permanent et pour faciliter l'analyse de données.

6.2.2.5. **Étape 5 - Inspection guidée du système SAM (modification du référencement sémantique)**

6.2.2.5.1. *Objectifs visés par l'étape*

Cette étape concerne l'évaluation de la modification du référencement sémantique à l'aide du système SAM. Les objectifs d'évaluation sont au nombre de deux :

- **Objectif 1** (étape 5). Le premier objectif est de vérifier, du point de vue de l'utilisateur, la pertinence sémantique des classes proposées pour la modification du référencement et de leurs caractéristiques.

- **Objectif 2** (étape 5). Le deuxième objectif est de valider l'utilité de la modification du référencement sémantique à l'aide du SAM mais aussi de préciser quel mode de modification (automatique *vs* assisté) est plus adéquat aux besoins des utilisateurs.

6.2.2.5.2. *Déroulement de l'étape*

Techniques d'évaluation

Nous avons utilisé la technique d'entretien et nous l'avons combinée à une inspection guidée de la fonctionnalité de modification du référencement.

Participants

Trois des six participants ont été convoqués pour cette étape d'évaluation : les participants P2, P3 et P4.

Procédure

La première tâche de chaque participant était d'explorer la fonctionnalité de modification du référencement sémantique. Cette exploration, réalisée sous notre supervision et notre guidage, s'est déroulée de la manière suivante. Premièrement, nous avons familiarisé les participants avec la notion du référencement sémantique (et celle d'UKI) et nous leur avons montré l'exemple d'un changement qui affecte le référencement (il s'agit du changement `MergeClasses` et ses changements additionnels). Ensuite, nous avons demandé aux participants d'explorer l'interface de modification du référencement, tout en leur donnant des explications sur les informations présentées, mais aussi sur la démarche à faire pour modifier les UKI affectés par le changement `MergeClasses`. Finalement, les participants, en prenant en compte les modifications suggérées par SAM, ont modifié les UKI affectés pour les rendre compatibles avec la nouvelle version V_{N+1}.

La deuxième tâche de chaque participant était de participer à un entretien composé d'un certain nombre des questions fondamentales et d'une série de questions secondaires.

Instrumentation

Les techniques et outils utilisés dans cette étape sont au nombre de trois : (1) l'exemple du changement `MergeClasses` et des UKI affectés ; (2) l'interface du SAM pour la modification des UKI affectés par le changement `MergeClasses`; (3) une liste des questions principales à aborder avec chaque participant.

a. Exemple du changement `MergeClasses` et des UKI affectés

Il s'agit du même exemple que celui déjà présenté dans la Figure V-20 : la fusion de `DesignerPedagogique` et `Professeur` dans une nouvelle classe, nommée `ConcepteurCours`, et ses changements additionnels. La seule différence réside dans la visualisation des changements à l'interface, dans le cas du SAM les changements qui affectent le référencement (ici les deux UKI référant aux classes fusionnées) sont soulignés à l'aide d'un code couleur.

b. Interface du SAM concernant la modification des UKI affectés

Un exemple du fonctionnement et de l'interface du système SAM, concernant la modification du référencement sémantique affecté par un changement de type `MergeClasses`, peut être consulté dans la Figure V-24.

c. Questions principales à aborder

La première question abordée demande aux participants de préciser si : (a) les classes proposées par SAM pour la modification des UKI sont significatives du point de vue sémantique ; (b) le fait de choisir parmi plusieurs classes est plus intéressant que celui d'avoir accès uniquement à la classe recommandée. La deuxième question permet aux participants d'indiquer si les informations fournies par SAM sont suffisamment riches pour permettre la modification des UKI affectés et de justifier leur réponse. Finalement, la troisième question sollicite des réponses concernant les modes de modification offerts (automatique *vs*. assisté) et questionne les sujets sur la dimension « conseiller » du système SAM.

6.2.2.5.3. *Collecte des données*

Les manipulations à l'interface, la verbalisation et les entretiens avec les participants ont été complètement enregistrés (image et son).

6.3 **Analyse des données d'évaluation**

Tous les résultats obtenus lors de l'évaluation des systèmes (étapes 2-5) sont des données qualitatives (verbalisations, discussions, notes, questions ouvertes et semi-ouvertes). Selon Miles et Huberman, (2003), l'analyse des données qualitatives s'élabore généralement au cours de trois étapes principales : la transcription des données recueillies, le codage des données et enfin l'analyse des données ainsi codées pour l'identification des occurrences ou des incidences dans les codes, qui pourront apporter des réponses aux objectifs formulés.

L'analyse des données par l'application de la méthode de Miles et Huberman (2003) a été limitée aux résultats obtenus à la suite de l'évaluation du CHB (*cf.* 6.3.1.). Étant donné la contrainte de temps et la redondance de l'exercice, nous avons limité l'analyse du SAM à quelques conclusions préliminaires (*cf. 6.3.2.*). Ces dernières découlent d'une écoute de la verbalisation des commentaires faite par les sujets lors des étapes 3 et 5 de l'expérimentation. Cependant, nous demeurons conscients qu'une analyse plus complète des données, fondée sur l'utilisation de la méthode de Miles et Huberman (2003), devrait être effectuée plus tard.

6.3.1. **Analyse des données provenant de l'évaluation du CHB**

Cette section présente la transcription des données issues de l'évaluation du CHB, leur codage et leur analyse, les conclusions finales portant sur l'utilité du système, mais aussi des propositions pour améliorer son utilisabilité.

6.3.1.1. **Transcription des données (le système CHB)**

Nous avons réalisé la transcription des données à l'aide du logiciel Excel, en utilisant le système orthographique du français et en respectant le plus possible la prononciation et la syntaxe des sujets. Cette transcription comprend la verbalisation des dyades (étapes 2) ainsi que la discussion des participants lors du focus-groupe (étape 4). Nous présentons cette transcription dans l'Appendice G et dans l'Appendice H.

6.3.1.2. **Codage des données (le système CHB)**

Le codage est une opération de marquage des segments de données avec des symboles descriptifs d'une caractéristique, les ***codes*** ou les ***catégories d'évaluation***, afin d'organiser les données brutes pour en faire émerger un sens. Un code correspond à une balise pour marquer une variable, un concept ou une valeur qui nous intéresse dans un corpus de données brutes. Selon Miles et Huberman (2003), il existe deux approches de codage de données : le codage *a priori* et le codage inductif. Dans le codage *a priori*, on décide à l'avance, selon des considérations théoriques par exemple, de différentes caractéristiques qu'on cherche dans le corpus des données. Dans le codage inductif, on commence par l'analyse d'un petit jeu de

données qu'on élargit ensuite en fonction de la théorie qui émerge, les codes étant établis pendant l'examen des données.

6.3.1.2.1. *Premier codage des données – codage inductif*

Pour l'analyse des données provenant de l'évaluation du CHB nous avons appliqué un compromis entre les deux types de codage : nous avons commencé par un codage inductif pour passer ensuite à un codage *a priori*. Comme point de départ du codage inductif, nous avons défini sept catégories générales (*cf.* Tableau VI-3), que nous avons utilisées ensuite pour marquer des sections dans le corpus des données.

Tableau VI-3. Codes généraux pour le codage inductif des étapes 2-5

Code	Explication du code
UTIL_CHG	section mettant en évidence l'utilité des changements complexes *vs* des changements élémentaires
COMPH	section mettant en évidence le développement de la compréhension des participants
PB_UTILISAB	section mettant en évidence les problèmes de compréhension causés par des problèmes d'utilisabilité
PB_COMPH	section mettant en évidence d'autres problèmes de compréhension (qui ne sont pas causés par de problèmes d'utilisabilité)
FONCT_U	section mettant en évidence l'utilité d'une fonctionnalité
FONCT_E, N	section mettant en évidence l'enrichissement (_E) ou l'ajout d'une fonctionnalité (_N)

Pendant qu'on codait à l'aide de ces sept catégories, nous avons identifié de nouvelles catégories (ou codes). Nous avons obtenu ainsi un ensemble complet des codes, dont la nature est justifiée par les objectifs d'évaluation du système CHB. Ces codes, présentés dans le paragraphe suivant, nous ont permis de regrouper et d'interpréter les verbalisations de chaque dyade afin d'en tirer les conclusions finales.

6.3.1.2.2. *Deuxième codage des données - codage déductif*

Dans le Tableau VI-4, le Tableau VI-5 ainsi que le Tableau VI-6 nous présentons les codes que nous avons utilisés pour le codage déductif des verbalisations. Nous avons défini ces codes en fonction des objectifs d'évaluation du système CHB. Nous les avons classifiés selon trois catégories principales :

- « connaître et comprendre la logique d'évolution de l'ontologie », catégorie associée à l'objectif 1 (étape 2);
- « formes de visualisation des changements », catégorie associée à l'objectif 3 (étape 2);
- « changements complexes *vs* changements élémentaires », catégorie associée à l'objectif 2 (étape 2).

Tableau VI-4. Codes associés à l'évaluation du CHB : catégorie CoEv – « connaître et comprendre l'évolution »

Type de réaction associée au code – catégorie CoEv	Code	Valeur
Énonciation du degré d'importance du fait de (a) savoir qu'une ontologie a évolué et (b) connaître comment elle a évolué (c.-à-d. connaître les changements) - du point de vue de l'acteur explorateur (exposant Ex) ou client (exposant Cl)	**(a) CoEv.Sa$^{Ex, Cl}$** **(b) CoEv.Con$^{Ex, Cl}$**	(+ signifie « positif », **+/-** « neutre », - signifie « négatif »)
Mention d'un avis sur le fait d'avoir une vue générale de l'ensemble de changements - sans l'aide du CHB (exposant S) ou avec l'aide du CHB (exposant A)	**CoEv.VueG$^{S, A}$**	(complète, incomplète) ; (certaine, incertaine) ; (facilitée, complexe/non facilitée)
Mention d'un avis correct sur la signifiance de l'évolution - sans l'aide du CHB (S) ou avec l'aide du CHB (A)	**CoEv.Sign ^{S,A}**	(avis)_(argumentation)
Identification d'un changement sans l'aide de CHB, ou la valeur *n*° (un numéro) permet de différencier un changement d'un autre dans un texte et les exposants E et C spécifient la validité du point de vue de changements élémentaires ou complexes	**CoEv.Chg^{S}**	(type, n°)_(+, +/-,-)^{E,C}; où le'+' signifie correct, le '+/-' en partie correcte, le'-' signifie incorrect.
Identification d'une relation de causalité (changement primaire/additionnel) sans l'aide de CHB, ou la valeur *rel* décrit la relation en question et *n*° la différentie dans le texte	**CoEv.RelC^{S}**	(rel, n°)_(+, +/-, -)
Exploration d'un changement ou d'une relation de causalité avec l'aide du CHB	**CoEv.Chg/Rel**	(type, n°)
Mention d'une (a) confusion ou (b) clarification dans la comparaison (exploration) de V_N et V_{N+1} effectuée pour identifier (ou comprendre) les changements appliqués - sans l'aide du CHB (S) ou avec l'aide du CHB (A)	**(a) CoEv.Conf^{S,A}** **(b) CoEv.Clarif^{S,A}**	(confusion/clarification)_(élém. générateur)
Mention d'une (a) incompréhension ou (b) compréhension d'un changement (*chg.n*°) ou d'une relation de causalité entre changements (*rel.n*°)	**(a) CoEv.Incomph** **(b) CoEv.Comph**	(chg.n°)_(élém. générateur) (chg.rel)_(élém. générateur)

Tableau VI-5. Codes associés à l'évaluation du CHB : catégorie Chg – « changements complexes *versus* élémentaires »

Type de réaction associée au code – catégorie Chg	Code	Valeur
Observation : Les changements élémentaires dont on parle ici réfèrent à des suites qui sont, en fait, des changements complexes (par exemple, on peut exprimer une fusion sous sa forme complexe, ou en utilisant l'effacement et l'ajout des classes).		
Mention d'une (a) incompréhension, ou respectivement (b) compréhension, de la « vraie » logique d'évolution (celle de l'ontologiste ayant fait évolué l'ontologie, cf. objectif 2, étape 2).	**(a) Chg.IL** **(b) Chg.CL**	_E(générée par un ou une suite de changements élémentaires) _C(générée par un changement complexe)
Mention d'une « fausse compréhension » de la logique d'évolution (c.-à-d., les sujets développent une compréhension, qu'ils considèrent correcte, qui n'est pas en accord avec la logique d'évolution, tel qu'elle a été appliquée par l'ontologiste ayant fait évoluer l'ontologie)	**Chg.FCo$^{N°}$**, où N° c'est un identifiant numerique	_E(*idem*) _C(*idem*)
Réparation d'une « fausse compréhension » de la logique d'évolution	**Chg.ReFCo$^{N°}$**	_E(*idem*) _C(*idem*)
Déduction de la vraie « intention formelle » (cf. objectif 2, étape 2)	**Chg.But**	(+, -)_E (généré par un ou une suite changements élémentaires) (+, -)_C (généré par un changement complexe)
Compréhension facilitée de la logique du processus d'évolution	**Chg.Fac**	(+, -)_E(*idem*) (+, -)_C(*idem*)
Mention du pouvoir d'expression (plus ou moins grand) de la visualisation des changements complexes comparée à celle de changements élémentaires	**Chg.Vis**	(+, -)_E(*idem*) (+, -)_C(*idem*)

Tableau VI-6. Codes associés à l'évaluation du CHB : catégorie Vis – « formes de visualisation de changements »

Type de comportement associé au code – catégorie Vis	Code	Valeur
Visualisation (exploration) de (a) changements ou (b) relation de causalité, accompagnée d'une consultation des versions d'ontologie	**(a) Vis.Chg.Onto** **(b) Vis.Pr/Add.Onto**	(**+** signifie « positif », **+/-** « optionnel », **-** signifie « négatif »)
Compréhension de la description de (a) changements ou (b) relation de causalité entre changements, description affichée à l'interface du CHB	**(a) Vis.Chg.Comp** **(b) Vis.Pr/Add.Comp**	(+, -)_Arg(arg); où *arg* signifie l'argument de l'analyste concernant la valeur positive ou négative accordée. Arg(OWL+) = description technique, axée OWL ; Arg(MOTPlus-) = description non liée au formalisme graphique de l'éditeur MOT+Onto ; Arg(sugg +, +/-,-) = description plus ou moins suggestive; Arg(autre).
Compréhension de la description de (a) changements ou (b) relation de causalité entre changements, affichée à l'interface du CHB, accompagnée d'une consultation des versions	**(a) Vis.Chg.Comp** **(b) Vis.Pr/Add.Comp**	(+,-)_Onto(+, +/-, -)

Dans le Tableau VI-7, nous illustrons comment nous avons utilisé ces codes pour marquer les données issues de la verbalisation des dyades. D'autres exemples de codage sont disponibles dans l'Appendice F et dans l'Appendice G.

Tableau VI-7. Exemples des extraits codés

Code	Exemples d'extraits codés (pour d'autres exemples, voir l'Appendice F et l'Appendice G
CoEv.Chg^{S}(remplacement, n°1.2)_(+/-)^{C} **CoEv.Conf**^{S}_(complexité des	« Là (en V_N), le `Pédagogue` est un animateur d'activités. Et là (en V_{N+1}), c'est le `Tuteur` qui fait ça. Donc, elle a changé `Pédagogue` pour `Tuteur`… mais qu'est-ce que c'est l'`Informateur` ?… ça devient compliqué, tout cela.»

changements appliqués)	*- Dyade formée par P2 et P6, le Test_B.1*
CoEv.VueG^{A}(complète, certaine, facilitée) **CoEv.Clarif^{A}**_(liste des changements générée par CHB)	« Du coût ça (le CHB) donne tous que … tous les changements apportés. […] En fait, c'est pas mal parce que ça (la liste des changements) aide à bien voir les changements. On n'a plus à chercher les changements sur le graphique (versions de l'ontologie en MOT+Onto) … ou on sait où chercher. » *- Dyade formée par P1 et P3, le Test_B.2*
Chg.IL_E	« - Là, la classe `DesignerIMS_LD` elle a été ajoutée. Tu vois, elle était là (le sujet montre son emplacement sur la V_N)… elle n'a pas été déplacée, mais rajoutée […] carrément effacée et rajoutée. - Effacer puis créer une nouvelle …c'est bizarre ça. » *- Dyade formée par P2 et P6, Test_B.2*
Chg.FCo$^{N°2.1}$ **Chg.ReFCo$^{N°2.1}$**	« Le `DesignerPedagogique` a été éliminé pour replacer ses deux sous-classes …» *- Dyade formée par P1 et P3, Test_B.2* «Mais en fait, elle (la classe `DesignerPedagogique`) est fusionnée avec `Prof`esseur […] ce sont ces deux classes-là qui sont fusionnées ... avec seulement `DesignerIMS_LD` qui est déplacée. » *- Dyade formée par P1 et P3, Test_B.3*
Chg.But(+)_C(fusion)	« Là, dans la deuxième fenêtre (la fenêtre VisualisationChangements_2) […] c'est une nouvelle action, c'est une fusion. Ah, OK … c'est ça qui nous donnent la raison! » *- Dyade formée par P2 et P6, Test_B.3*
Vis.Chg.Comp(-)_(OWL+, sugg -)	"Ensuite c'est *addClass* `ConcepteurCours` *to PropertyDomain* … Ah, ça je ne comprends pas, trop difficile. […] C'est la façon dans laquelle c'est écrit. » *- Dyade formée par P2 et P6, Test_B.2*
Vis.Chg.Comp(+)_Onto(+)	« - […] c'est *AddDisjointClass* `Étudiant Collège` au `EtudiantUniversitaire`. […] OK. C'est l'ajout d'une disjonction. C'est qu'ils ont rajouté ici (le sujet encercle l'axiome sur la version V_{N+1}) le lien de disjonction. » *- Dyade formée par P1 et P3, Test_B.2*

6.3.1.3. **Analyse des données (le système CHB)**

Avant de présenter l'interprétation des données qualitatives, rappelons nos objectifs d'évaluation du système CHB. Le **premier objectif**, dont la catégorie de codes associée est la **CoEv** (connaître et comprendre l'évolution), est celui de déterminer le degré d'influence du CHB sur la compréhension du processus d'évolution d'une ontologie. Le **deuxième objectif**, dont la catégorie de code associée est **Chg** (changements complexes *vs* élémentaires), est celui de déterminer dans quelle mesure, le fait de visualiser des changements complexes, en plus des changements élémentaires, permet une meilleure compréhension de la logique d'évolution ainsi que de '*l'intention formelle'* des ontologistes ayant modifié l'ontologie. Enfin, le **troisième objectif**, dont la catégorie de codes associée est la **Vis** (formes de visualisation des changements), est celui d'identifier les principaux problèmes d'utilisabilité observés par et chez les participants et de trouver des solutions à cet effet.

Pour interpréter les données, nous les avons premièrement organisées en fonction des codes associés :

1. Nous avons regardé les codes effectivement utilisés dans le codage des données (tous les codes prédéfinis n'ont pas été utilisés pour coder les verbalisations). Un code utilisé au moins une fois a été retenu pour l'interprétation des données. De même, s'il existe une justification importante associée à un certain code, elle a aussi été retenue ;

2. Pour chaque code utilisé, nous avons compté ensuite les occurrences par dyade, c'est-à-dire le nombre de fois que le même code, avec la même valeur (+, +/-, -), apparaît dans le codage de la verbalisation d'une même dyade ;

3. Finalement, nous avons organisé les codes en fonction de l'information qu'on peut en retirer pour répondre aux objectifs d'évaluation, comme présenté dans les sections suivantes.

6.3.1.3.1. *Catégorie de codes – CoEv 'connaître et comprendre l'évolution'*

Nécessité de connaître l'évolution

La Tableau VI-8 présente l'ensemble des codes parlant de la nécessité de connaître les changements apportés pour passer d'une version d'ontologie à une autre, et cela du point de vue d'un acteur explorateur ou client de l'ontologie[55]. Pour chaque code, le nombre d'occurrences par dyade est explicité à coté de la valeur qui lui est associée (p.ex. on a retrouvé trois fois le code $CoEv.Con^{Ex}$(+) dans la verbalisation de la dyade 1). L'indicatif *Arg* indique si la dyade en question a ou non justifié ses réponses.

Tableau VI-8 : Nécessité de connaître l'évolution

Codes	Dyade 1 (P2 et P6)		Dyade 2 (P1 et P3)		Dyade 3 (P4 et P5)	
$CoEv.Sa^{Ex}$	(+)_1	Arg(+)	(+/-)_1	Arg (-)	(+/-)_1	Arg (+/-)
$CoEv.Sa^{Cl}$	(+)_3		(+)_1		(+)_1	
$CoEv.Con^{Ex}$	(+)_3		(+)_1 et (+/-)_1		(+)_1 et (+/-)_1	
$CoEv.Con^{Cl}$	(+)_4		(+)_1		(+)_2	

Nous remarquons que les dyades 2 et 3 affirment qu'il n'est pas vraiment nécessaire, pour un explorateur de l'ontologie, de savoir qu'une ontologie a été modifiée, ni de connaître, ou de comprendre les changements apportés. Cependant, les sujets de chacune de ces dyades avancent un avis différent : tandis qu'un pense qu'il est nécessaire pour une exploration de connaître les changements (par exemple, pour voir s'il est d'accord avec la nouvelle vue du domaine), un autre affirme que c'est seulement important, sans que cela soit vraiment essentiel. En ce qui est d'un client de l'ontologie, les dyades 2 et 3 affirment qu'il est essentiel de connaître comment l'ontologie a évolué afin de décider si : (a) dans le contexte du référencement sémantique, les références restent valides et (b) dans le contexte de la conception pédagogique, le concepteur est ou non d'accord avec la nouvelle conceptualisation, sachant que l'évolution d'une ontologie peut « même défaire le projet de conception » (*cf.* dyade 3).

[55] La définition de ce qu'est un acteur explorateur ou client a été donnée dans le paragraphe 6.2.2.2.2.

Concernant la première dyade, on observe une discussion plus poussée au sujet de la nécessité de comprendre l'évolution, comme l'indique d'ailleurs le nombre d'occurrences par code. Ici, les deux sujets soutiennent la nécessité de savoir et de comprendre, que ce soit pour un acteur explorateur ou client de l'ontologie, mais ils se concentrent toutefois sur l'acteur client. Ils soulignent que, d'un point de vue objectif, cela est nécessaire puisque le client doit tenir compte des changements apportés, s'il veut utiliser la nouvelle version de l'ontologie, qui est « le domaine représenté explicitement et qu'on est obligé de respecter » (*cf.* dyade 1). D'un point de vue plus subjectif, il est nécessaire pour un client de connaître les changements afin de pouvoir se positionner par rapport à la nouvelle vue du domaine, ce qui rejoint un des avis de dyades 2 et 3.

Pour résumer, on peut affirmer que la totalité des sujets a validé la nécessité de connaître les changements apportés aux versions d'ontologie, du point de vue d'un acteur client, et certains d'entre eux (4/6), même pour un acteur explorateur de l'ontologie.

Développement de la compréhension de l'évolution sans le CHB

Le Tableau VI-9 reflète de quelle manière les sujets développent leur compréhension de l'évolution, **sans l'aide** de la fonctionnalité d'identification et de visualisation des changements.

Pour chaque code, le nombre d'occurrences par dyade est explicité par un nombre, à côté du code (p.ex. pour la dyade 1, on a retrouvé deux fois le code CoEv.VueG^{S}). En ce qui concerne le code CoEv.Chg^{S}, nous tenons à rappeler que l'indicatif *Chg* réfère au type de changement et les indicatifs *E* et *C* permettent d'établir la validité du changement ('+' pour correct, '+/-' en partie correct, '-' pour incorrect) du point de vue de changements élémentaires et/ou complexes. Principalement, nous voulons montrer ici que, bien qu'un changement identifié puisse être correct si on le regarde dans une perspective élémentaire, par exemple l'effacement de la classe `Professeur`, il est incorrect si on le regarde dans une perspective complexe, par exemple la classe `Professeur` a été fusionnée avec la classe `DesignerPedagogique`. Si, pour un changement identifié, on retrouve la valeur (+) pour l'indicatif E et (-) pour le C, nous sommes dans la situation présentée ci-dessus. Les autres situations ne concernent qu'un seul indicatif, que ce soit E ou C.

Tableau VI-9 Développement de la compréhension de l'évolution, sans le CHB

Code	Dyade 1 (P2 et P6)			Dyade 2 (P1 et P3)			Dyade 3 (P4 et P5)		
CoEv.Sign^S	(significative)_1			(significative)_1			(significative)_1		
	Arg (suppression et modification des classes)								
CoEv.Conf^S	_2 (complexité ontologies et changements apportés)			_0			_1 (complexité changements apportés)		
CoEv.VueG^S(incomplète, incertaine, complexe)	_2			_1			_1		
CoEv.Chg^S (classe ou propriété)	Le nombre total de changements (pour VisualisationChangements_02) est 18 dont 13 complexes et 5 élémentaires								
	_5(au total)	E	C	_4(au total)	E	C	_6(au total)	E	C
- C*hgE*: ajout	_1	(-)		_1	(+)	(-)	_0		
- *ChgE* : effacement	_2	(+) (+)	(-)	_0			_2	(+) (+)	(-) (-)
- *ChgC* : remplacement	_2		(-) (+/-)	_1		(-)	_1		(-)
- *ChgC* : déplacement	_0			_1		(+/-)	_1		(+)
- *ChgC* : division	_0			_1		(+/-)	_1		(-)
- *ChgC* : fusion	_0			_0			_1		(-)

En regardant le tableau ci-dessus, nous observons que toutes les dyades ont identifié correctement le fait que l'ontologie a évolué d'une manière significative, en argumentant même que c'est parce qu'il y a eu des modifications et des effacements des classes. Tous les sujets mentionnent cependant que, même s'ils croient à une évolution significative, ils ne peuvent avoir qu'une vue très partielle de l'ensemble des changements apportés pour passer de V_N à V_{N+1}. Ils affirment que, même s'ils peuvent identifier quelques changements, ils ne peuvent ni être certains qu'ils soient corrects, ni identifier l'ensemble de changements dans son intégralité. Deux dyades manifestent également une confusion lors de l'identification des changements par eux-mêmes (*cf.* Test B.1), en la justifiant par le fait que les ontologies et les changements apportés sont trop complexes.

D'autres résultats viennent appuyer le fait que l'identification des changements est trop complexe pour qu'elle soit effectuée par les sujets eux-mêmes. Ainsi :

- les sujets de la dyade 1 identifient seulement cinq changements du total des dix-huit apportés (treize complexes et cinq élémentaires). Parmi ces cinq changements, on retrouve trois élémentaires (type : ajout et effacement) et deux complexes (type : remplacement). Concernant les changements élémentaires identifiés, un est correct et un incorrect, et le troisième (effacement de la classe `Profeseur`), même s'il est correct du point de vue élémentaire (la classe n'existe plus dans la V_{N+1}) est totalement incorrect si on le regarde dans sa perspective complexe (la classe `Professeur` n'as pas été effacée mais fusionnée avec une autre classe). Concernant les changements complexes, un est incorrect et un partiellement correct au sens que, bien que les sujets aient identifié une des caractéristiques du changement de fusion – « le `DesignerPédagogique` […] est devenu `ConcepteurCours` » –, ils n'ont pas pu identifier effectivement le changement.

- les sujets de la dyade 2 identifient seulement quatre changements, dont trois complexes (un incorrect et deux partiellement corrects) et un élémentaire. Ce dernier (l'ajout de la classe `Informateur`) est aussi incorrect si on le regarde dans une perspective complexe (comme la division de la classe `Pedagogue` en `Tuteur` et `Informateur`).

- les sujets de la dyade 3 identifient six changements du total de dix-huit apportés. Parmi les changements identifiés, quatre sont complexes, dont uniquement un qui est correct, et les deux autres sont élémentaires. Comme pour les autres dyades, ces deux derniers changements, bien qu'ils soient corrects du point de vue élémentaire, ne correspondent pas aux vrais changements complexes apportés.

En conclusion, nous pouvons affirmer que, si dans le contexte des versions d'ontologie de petite taille (approximativement 22 classes et 8 propriétés), tous les sujets ont montré une grande difficulté à identifier des changements qui soient corrects, dans le contexte des ontologies courantes cela devient encore plus difficile, voire impossible. Ceci est confirmé d'ailleurs par le fait que les sujets ont passé plus de quinze minutes à explorer les deux

versions d'ontologie, sans pour autant réussir à identifier plus de cinq changements en moyenne, dont la plupart se sont avérés incorrects.

Un autre point que nous aimerions souligner est que, même si les sujets identifient un certain nombre de changements élémentaires, la majorité de ceux-ci ne correspondent pas aux vrais changements accomplis par l'ontologiste, lors de l'évolution de l'ontologie. Ceci peut alors conduire au développement de ce qu'on appelle une 'fausse compréhension', c'est-à-dire une compréhension que les sujets jugent correcte, mais qui ne correspond pas au processus d'évolution de V_N à V_{N+1}.

Si on ajoute à tout cela la confusion observée chez les sujets lors de la comparaison entre V_N et V_{N+1}, on peut affirmer que, sans l'aide d'un support pour les guider dans l'identification, voire la visualisation des changements apportés aux versions d'ontologie, les sujets ne peuvent développer qu'une compréhension très partielle, et assez incorrecte, du processus d'évolution. Ils ne peuvent pas préciser l'ensemble des changements apportés à l'ontologie, ni valider les quelques changements qu'ils avaient identifiés par eux-mêmes.

Développement de la compréhension de l'évolution avec le CHB

Le Tableau VI-10 traite de la manière dont les sujets développent leur compréhension de l'évolution, **avec l'aide** de la fonctionnalité d'identification et de visualisation des changements. Dans ce tableau, nous avons réuni les codes qui réfèrent à la mention d'une compréhension (ou d'une clarification) grâce au système CHB. Dans le Tableau VI-11, nous avons regroupé l'avis des sujets quant à l'utilité même de ce système.

En regardant le Tableau VI-10, nous observons premièrement que les sujets conservent leur avis au sujet de la signifiance de l'évolution, à savoir : l'ontologie a évolué d'une manière significative puisqu'il y a eu des suppressions et des modifications de classes. Les sujets affirment également qu'avec l'aide du système CHB ils peuvent avoir une vue complète et systématique de l'évolution (« la liste nous donne tous les changements apportés **[**…**]** ça nous aide à bien voir les changements », *cf.* dyade 2) ainsi que la conviction que cette vue présente une information véridique au sujet des changements apportés (« [...] et tu ne dois plus te demander si les changements sont ou non corrects », *cf.* dyade 1).

Tableau VI-10. Développement de la compréhension de l'évolution, avec le CHB

Code	Dyade 1 (P2 et P6)	Dyade 2 (P1 et P3)	Dyade 3 (P4 et P5)
CoEv.Sign^{A}	(significative)_1	(significative)_1	(significative)_1
	Arg (suppression et modification des classes)		
CoEv.Clarif^{A}_(CHB)	_1	_1	_0
CoEv.VueG^{A}(complète, certaine, facilitée)	_2	_1	_1
CoEv.Chg^{A} (classe ou propriété)	Changements explorés lors du Test B.2 et le Test B.3		
	≈ 11, dont 5 complexes	≈ 15, dont 6 complexes	≈ 9, dont 4 complexes
CoEv. Comph_(CHB)	_5 (au total)	_11 (au total)	_4 (au total)
- ajout et/ou effacement	_1(axiome)	_2(axiome et classe)	_0
- transfert de classe et et/ou propriétés	_1	_3	_1
- division et/ou fusion	_2	_2	_1
- causalité Pr/Add	_1	_4	_2

Tableau VI-11. Mention de l'utilité du CHB pour la compréhension de l'évolution

Code	Dyade 1	Dyade 2	Dyade 3
FONCT_U(identification et visualisation des changements avec le CHB)	_3	_2	_1

La fonctionnalité d'identification et de visualisation des changements permet le développement, chez les sujets, d'une bonne compréhension de l'évolution de l'ontologie et leur facilite l'acquisition d'importants gains cognitifs. Ainsi :

- Parmi les onze changements explorés, les sujets de la dyade 1 mentionnent cinq compréhensions qu'il n'ait pas pu avoir sans le CHB : le transfert d'une classe[56] et le fait que c'est un changement additionnel (c.-à-d. une conséquence d'un autre changement) ou encore la compréhension de changements plus complexes comme la

[56] Ce changement est équivalent, syntaxiquement et sémantiquement, avec le changement qui modifie la superclasse d'une classe dans la taxonomie (le `ModifySuperClass`).

division et la fusion des classes. Pour donner un exemple, prenons un extrait de la verbalisation de la dyade 1 : « Il y a des changements qu'on n'a pas vus comme ça, à l'œil. […] (par exemple, pour la classe `AnimateurActivité`) on a changé sa classe supérieure, de la classe `Pédagogue` à la classe `Tuteur`, parce que `Pédagogue` a été […] changé pour `Tuteur` et `Informateur` ».

- Les sujets de la dyade 2 mentionnent sept changements qu'ils ont compris pendant qu'ils consultaient la fenêtre de visualisation générée par le système CHB : des changements d'ajout d'axiomes et des changements plus complexes, comme ceux de division et de fusion de classes, accompagnés de leurs conséquences (p.ex. le transfert de sous-classes et des propriétés). Ils affirment que « ça veut dire que cela (la propriété `encadreEtudiants`) est déplacée de `Pédagogue` à `Tuteur`. […] C'est exactement ça, c'est d'un déplacement du domaine ici qu'on parle ». Les sujets ont particulièrement apprécié le fait qu'ils peuvent voir les relations de causalité entre les changements primaires et les changements additionnels (on observe quatre mentions à ce sujet).

- Les sujets de la dyade 3, après avoir exploré neuf changements à l'aide du CHB, affirment avoir mieux compris un changement d'ajout d'un axiome, mais aussi celui de division d'une classe et le transfert de ces sous-classes. Par contre, ils ont de la difficulté à percevoir le changement de fusion, comme nous allons le montrer plus bas. Ils saisissent cependant très bien le sens de ce que sont les changements additionnels : « ce qu'on voit ici (dans fenêtre de visualisation des changements), c'est comme une hiérarchie des changements, parce que ceux-ci sont des modifications principales et les autres en sont des conséquences […] apparues pour répondre à la question de quoi faire avec les sous-classes de `Pédagogue`, par exemple ».

En conclusion, le CHB, plus précisément sa fonctionnalité d'identification et de visualisation des changements apportés aux versions d'ontologie, offre aux sujets la possibilité de mieux connaître et comprendre l'évolution d'une ontologie (objectif 1 de l'étape 2), étant donné qu'elle :

- **Facilite la compréhension**. Par le fait que le CHB offre aux sujets l'accès à une vue complète sur les changements, il les aide à saisir tant l'ampleur du processus d'évolution. De plus, la confusion montrée par les sujets lors de l'identification des changements par eux-mêmes (*cf.* Test B.1), n'est plus du tout observable quand ils utilisent le système CHB. Au contraire, les sujets mentionnent que le fait d'avoir accès aux changements leur permet de clarifier beaucoup des questionnements qu'ils se posaient auparavant, quand la fenêtre de visualisation des changements n'était pas disponible.

- **Certifie la compréhension**. La fenêtre d'identification et de visualisation des changements (élémentaires et complexes) a l'avantage de présenter les vrais changements apportés et, ainsi, de permettre, chez les sujets, le développement d'une compréhension correcte, en accord avec le processus d'évolution et avec l'intention formelle de l'ontologiste ayant accompli ce processus.

- **Structurer la compréhension**. Comme nous l'avons déjà discuté ci-dessus, la visualisation des changements avec le CHB conduit les sujets à mieux comprendre un grand nombre de changements, mais aussi à mieux structurer leur compréhension. Ce dernier aspect est du principalement au fait que les changements sont ordonnés selon la relation de causalité de type primaire/additionnel.

Pour appuyer ces conclusions, et valider l'utilité du CHB, nous observons que la totalité des sujets (*cf.* Tableau VI-11) soutient la pertinence de la fonctionnalité d'identification et visualisation des changements et affirme, plus d'une fois, que cette fonctionnalité est « cruciale », « clairement nécessaire » ou « très importante».

6.3.1.3.2. *Catégorie de codes – Chg 'changements complexes vs élémentaires'*

Dans cette section, nous présentons la manière dont les sujets saisissent la logique et l'intention formelle[57] des ontologistes ayant fait évoluer l'ontologie. Ceci, selon que le CHB présente aux sujets les changements complexes sous leur vraie forme, c'est-à-dire leur forme

[57] Le terme d'intention formelle a été défini dans la section 6.2.2.2.1.

complexe, ou sous la forme d'une suite des changements élémentaires. Dans le Tableau VI-12, le Tableau VI-13 et le Tableau VI-14, nous résumons les résultats obtenus après le codage de verbalisation avec les codes provenant de la catégorie *Chg* (changements complexes *vs* élémentaires), le nombre d'occurrences par dyade, pour chaque code, est explicité par un nombre.

Tableau VI-12 (In)compréhension de la logique d'évolution

Code	Dyade 1 (P2 et P6)	Dyade 2 (P1 et P3)	Dyade 3 (P4 et P5)
Chg.IL_E	_2	_3	_1
Chg.CL_C	_3	_1	_0
Chg.But(-)_E	_2	_1	_2
Chg.But(+)_C	_2	_1	_1

Tableau VI-13 Développement/réparation d'une fausse compréhension

Code	Dyade 1 (P2 et P6)	Dyade 2 (P1 et P3)	Dyade 3 (P4 et P5)
Chg.FCo_E	_n°1.1 _n°1.2	_n°2.1 _n°2.2	_n°3.1
Chg.ReFCo_C	_1 (n°1.2)	_1 (n°2.2)	_0

Tableau VI-14 Facilitation de la compréhension 'changements complexes vs élémentaires'

Code	Dyade 1 (P2 et P6)	Dyade 2 (P1 et P3)	Dyade 3 (P4 et P5)
Chg.Fac(-)_E	_1	_3	_0
Chg.Fac(+)_C	_2	_2	_0
Chg.Vis(-)_E	_0	_1	_0
Chg.Vis(+)_C	_2	_1	_0

Avant de discuter les résultats de cette catégorie, rappelons que l'évaluation du système CHB s'était déroulée en plusieurs sous étapes (*cf.* Section 6.2.2.2.2.) :

- Le TestB.2 (et la fenêtre VisualisationChangements_1) qui présentait : (a) la visualisation d'un changement complexe (la division de la classe `Pedagogue`) et de ses changements additionnels, sous sa vraie forme complexe ; (b) la visualisation d'un autre changement complexe (la fusion des classes `DesignerPedaogique` et `Professeur`), et leurs changements additionnels, sous une forme élémentaire.

- Le test B.3 (et la fenêtre VisualisationChangements_2) qui, pour le même couple de versions V_N et V_{N+1}, présentait les deux changements complexes (la division et la fusion), et leurs changements additionnels, directement dans leur format complexe.

Précisons également que les changements élémentaires dont on parle ici référent à des suites qui sont, en fait, des changements complexes (par exemple, on peut exprimer une fusion en utilisant l'effacement et l'ajout des classes, comme dans la fenêtre VisualisationChangements_1).

Dans ce contexte, le Tableau VI-12 met en évidence que l'utilisation des changements complexes et/ou élémentaires[58] a un effet différent sur la compréhension de la logique d'évolution et de l'intention formelle de l'ontologiste ayant fait évoluer l'ontologie. Le Tableau VI-13 fait ressortir les problèmes de 'fausse compréhension', montrés par les sujets après avoir consulté des changements élémentaires en lieu des changements complexes.

Comprendre la logique d'évolution

Après la consultation de la fenêtre VisualisationChangements_1, on observe chez les sujets une incompréhension de la logique de changements apportés à V_N pour obtenir V_{N+1}, comme le montre les résultats du Tableau VI-12. La plupart du temps, cette incompréhension est produite par des ensembles de changements élémentaires n'ayant aucune signification pour les sujets, ou ayant une signification qui est loin de la vraie intention formelle sous-jacente à l'évolution. Prenons l'exemple le plus répandu, celui de l'effacement et de l'ajout d'une même classe pour exprimer, en fait, un transfert de classe. Dans cette situation, les sujets se sont montrés extrêmement confus. Ils disent que « la classe `Designer_IMS_LD` n'a pas été déplacée, elle a été rajoutée, carrément effacée et rajoutée. C'est bizarre ça » (*cf.*

[58] L'utilisation des changements complexes et/ou élémentaires pour exprimer la même chose.

dyade 1), ou ils se questionnent « Il n'y a pas une erreur ici puisque `Designer_IMS_LD` est supprimé, mais il est toujours là (dans V_{N+1}) ? », (*cf.* dyade 2). Cette difficulté de saisir la logique de l'évolution est mise aussi en évidence dans le Tableau VI-14, où on observe que la plupart des sujets déplorent fortement le fait d'avoir accès uniquement aux suites de changements élémentaires : « on est obligé de laisser des choses en suspens, parce qu'il y a d'autres changements qui suivent plus bas … c'est comme s'ils sont traités en deux temps, ce qui rend la compréhension assez difficile » (*cf.* dyade 2).

Par contre, dès qu'ils ont eu accès aux changements complexes, présentés dans la fenêtre VisualisationChangements_2, la plupart des sujets ont été en mesure de bien appréhender la logique de l'évolution, en affirmant qu'on « comprend vraiment mieux les changements qui ont été faits » (*cf.* dyade 1). Ils ont même été capables de faire la différence entre la qualité des changements complexes, contrairement à celle des changements élémentaires : « […] pourquoi l'avoir fait d'une telle manière … l'effacement et l'ajout d'une même classe ? Pourquoi pas comme ici, où on transfère les sous-classes (dans le contexte du changement de division)? ». De plus, les sujets indiquent formellement le fait que l'accès à la fenêtre VisualisationChangements_2 leur a beaucoup facilité la compréhension de l'évolution (*cf.* Tableau VI-14).

Prendre conscience de l'intention formelle sous-jacente à l'évolution

Ce paragraphe traite de la capacité des sujets à saisir l'intention formelle de l'ontologiste ayant modifié l'ontologie. Comme le montre le Tableau VI-12, cette intention formelle n'est pas saisie par les sujets, quand ils sont confrontés uniquement aux changements élémentaires. Ils affirment que « là, justement, c'est difficile de savoir le but, ce n'est pas évident de comprendre le rationnel, l'intention qui est derrière tout ça », *cf.* dyade 1.

Par contre, après avoir consulté la fenêtre VisualisationChangements_2, qui montre un changement complexe de fusion, exprimé auparavant sous la forme d'un certain nombre d'effacements et d'ajouts, les sujets sont capables de mieux saisir l'intention formelle sous-jacente à l'évolution. Ils affirment que « c'est une nouvelle action ici, c'est une fusion. […] c'est ça qui nous donne la raison », *cf.* dyade 1, ou encore que « on a une `MergeClasses` ici, qu'on n'avait pas auparavant. […] c'est plus précis et plus significatif aussi. C'est bien plus

clair maintenant ce qu'ils (les ontologistes) ont voulu faire pour changer l'ontologie », *cf.* dyade 2.

Nous observons néanmoins que, bien que les sujets de la dyade 3 se rendent compte du changement de fusion, ils ne peuvent pas faire la différence entre la « fausse » intention (effacements et ajouts des classes) et la « vraie » intention formelle (fusion des classes). Ils considèrent qu'il s'agit de deux intentions différentes, même si « le résultat de l'évolution est identique ». Nous attribuons cette confusion à une 'fausse compréhension', que les sujets ont développée lors de la consultation de la fenêtre VisualisationChangements_1, et qu'ils ne peuvent plus remettre en question par la suite.

Développer une fausse compréhension de l'évolution

Nous avons déjà parlé de ce qu'on appelle une 'fausse compréhension' de l'évolution de l'ontologie (p. 216). Les résultats présentés dans le Tableau VI-13 reviennent sur cet aspect. Ces résultats montrent que, si le CHB présente aux utilisateurs les changements sous une forme élémentaire, ceux-ci peuvent développer de 'fausses compréhensions' de l'évolution. Un exemple d'une telle compréhension est celui de la dyade 2, qui affirme que « le `DesignerPedagogique` a été éliminé pour replacer ses deux sous-classes », ou encore celle de la dyade 3, qui souligne que « la division de `Professeur` a entraîné plusieurs changements additionnels ». Ceci, en sachant qu'en vérité les deux classes ont été fusionnées.

Cependant, dès qu'ils ont accès aux changements complexes, fournis dans la fenêtre VisualisationChangements_2, la plupart des sujets sont capables de remettre en question leur 'fausse compréhension' et d'en acquérir une qui est correcte, comme dans l'exemple suivant : « Avant on avait pensé que l'on avait supprimé (la classe `DesignerPedagogique`) et rajouté les sous-classes par la suite. Mais en fait, elle est fusionnée avec `Professeur` […] avec `Designer_IMS_LD` qui est déplacée », *cf.* dyade 1. Pour appuyer cette conclusion, le Tableau VI-14 nous montre que les sujets ont trouvé cette deuxième fenêtre « plus parlante » pour la compréhension de l'évolution puisque « c'est plus condensé, les changements sont regroupés sous une seule signification ».

Parlons maintenant du cas de la troisième dyade, dont les sujets, même confrontés aux changements complexes, ne peuvent pas remettre en question leur 'fausse compréhension' :

« là (dans la fenêtre VisualisationChangements_1), tu fais la division de la classe `Professeur` et ensuite tu refais **[**…**]** là (dans la fenêtre VisualisationChangements_1), c'est la fusion, on ne la comprend pas ». Ce problème valide encore plus la nécessité de présenter aux utilisateurs les vrais changements complexes apportés à l'ontologie, non pas se limiter à une liste des changements élémentaires qui n'ont pas la même signification pour les sujets.

Conclusions de la section 'changements complexes vs élémentaires'

Dans ce paragraphe, nous résumons les résultats concernant le deuxième objectif d'évaluation du système CHB, objectif qui visait à déterminer l'avantage des changements complexes par rapport aux changements élémentaires. Ces avantages sont au nombre de trois :

- **Clarification de la logique d'évolution**. Les changements complexes permettent aux utilisateurs du CHB de se rendre compte de la logique d'évolution et leur facilitent la compréhension de 'vrais' changements apportés à l'ontologie. À l'inverse, les changements élémentaires créent des problèmes de compréhension ainsi qu'une confusion chez les utilisateurs puisque ces derniers ne peuvent ni suivre logiquement, ni être d'accord avec le déroulement de l'évolution qui leur est présentée.

- **Appréhension de l'intention formelle**. Les changements complexes, contrairement aux changements élémentaires[59], permettent aux utilisateurs d'appréhender l'intention formelle de l'ontologiste. Toutefois, bien que la plupart des sujets aient affirmé que les changements complexes explicitent cette intention formelle, ils ont déploré toutefois le fait qu'ils ne peuvent pas spécifier « le pourquoi … *the meaning that is in the world*, dans la tête de gens »[60].

- **Développement d'une 'vraie' compréhension**. Finalement, les changements élémentaires peuvent produire chez les utilisateurs du CHB une 'fausse'

59 Nous parlons ici des changements élémentaires qui expriment des modifications complexes.
[60] Du point de vue de la machine, cette pensée interne ne peut être « comprise » sans l'aide de l'humain. Et comme notre contexte est celui du Web sémantique et des ontologies formelles, le fait de saisir cette pensée interne ne fait pas encore partie de nos tâches.

compréhension, les empêchant de bien comprendre l'évolution. Pour éviter cela, les changements complexes deviennent ici indispensables au développement d'une bonne compréhension du processus d'évolution d'une ontologie.

6.3.1.3.3. *Catégorie de codes – Vis 'formes de visualisation des changements'*

Lors de l'évaluation du CHB, notre intérêt s'est tourné vers son utilité plus que vers son utilisabilité. Cependant, dans ce paragraphe nous essayons d'identifier quelques problèmes d'utilisabilité ayant une influence négative sur le développement de la compréhension chez les sujets. Ces problèmes sont principalement reliés à la manière dont les changements sont présentés à l'interface, comme nous le résumons dans le Tableau VI-15.

Tableau VI-15. Présentation de changements et problèmes d'utilisabilité

Code	Dyade 1 (P2 et P6)	Dyade 2 (P1 et P3)	Dyade 3 (P4 et P5)
Vis.Chg.Onto(+)	_3	_3	_2
Vis.Chg.Comp(-) _Arg(OWL+, sugg-)	_5	_4	_2
Vis.Chg.Comp(+)_Onto(+)	_1	_2	_0

Lors de l'évaluation du CHB, les sujets ont ressenti le besoin d'explorer la liste des changements en faisant des allers-retours entre celle-ci et les versions V_N et V_{N+1} de l'ontologie. Bien qu'on puisse dire alors que ces versions se sont avérées un outil précieux pour la compréhension de l'évolution, ceci indique toutefois un déficit de l'utilisabilité du CHB, particulièrement au niveau de l'interface. Cette conclusion est d'ailleurs validée par les deux aspects suivants :

- **Incompréhension générée par la description trop technique des changements.** Les sujets ont manifesté une incompréhension de la description des changements, affichée par le CHB, étant donné que celle-ci est trop axée sur le formalisme OC (*Ontology Change*) de représentation de changements et sur le formalisme OWL de représentation des ontologies. Les sujets ont alors « du mal à comprendre la description des changements » à cause « de la façon dans laquelle c'est écrit, trop technique ». Ils manifestent leur confusion en disant : « *DeletePropertyDomain*, ça

veut dire que la propriété domaine qui s'appelle … enfin, qu'est-ce que c'est ça, je ne comprends pas ? ».

- **Compréhension accompagnée d'une consultation de versions d'ontologie**. En accompagnant cependant la consultation de l'interface du CHB par une lecture des versions d'ontologie, les sujets ont compris finalement la description des changements. Un exemple concluant est celui fourni par le dialogue suivant : « - C'est une *AddDisjointClass* - C'est ça quoi (le sujet P3 montre le lien de disjonction sur la version V_{N+1}) […] c'est une classe disjointe. - OK. C'est qu'ils ont rajouté ici (le sujet P1 encercle l'axiome sur la version V_{N+1}) le lien de disjonction ». Ce n'est alors pas la compréhension de l'évolution qui est un problème ici, mais plus le manque d'utilisabilité du CHB et ses faibles caractéristiques d'interface.

6.3.1.3.4. *Catégorie de codes – FONCT_N, FONCT_E 'fonctionnalités à ajouter ou enrichir' et des résultats du focus-groupe*

Dans cette section, nous présentons les nouvelles fonctionnalités à ajouter au CHB pour rendre le système plus riche sémantiquement, mais aussi plus utilisable. Pour cela, nous avons regroupé dans le Tableau VI-16 et le Tableau VI-17 toutes les suggestions exprimées par les sujets lors de l'évaluation.

Enrichissement des fonctionnalités du CHB

Tableau VI-16 Solutions pour l'enrichissement des fonctionnalités existantes dans le CHB

Enrichissement de fonctionnalités du CHB	Occurrence
Fonction de journalisation - Annotation des changements	Verbalisation dyade 1 et 3 Focus-groupe
Visualisation de la liste des changements - Guide de lecture - Liste des changements en code couleur	 Verbalisation dyade 1 et 2 Verbalisation dyade 1

Annotation des changements. Les sujets ont exprimé le besoin de connaître 'le pourquoi' de l'évolution, la motivation des ontologistes l'ayant accomplie. Pour cela, ils ont proposé d'introduire une fonctionnalité d'annotation des changements, permettant : (a) l'ajout

des commentaires textuels, introduits par les ontologistes lors du processus d'évolution, pour expliquer leur motivation; (b) la clarification de la signification d'une nouvelle classe (ou d'une classe qui a été modifiée); (c) l'explicitation de l'identité des ontologistes et leur expertise dans le domaine conceptualisé par l'ontologie; (d) la description du contexte d'occurrence des changements.

Visualisation de la liste des changements. Pour ce qui est de la visualisation des changements sous la forme d'une liste hiérarchique, comme générée actuellement par le CHB, les sujets ont mentionné deux améliorations possibles : (a) intégrer un guide de lecture, accessible à partir de l'interface, pour expliquer les termes techniques utilisés ; (b) souligner le type de changements apportés, en utilisant, par exemple, un code de couleur[61].

Identification des nouvelles fonctionnalités à ajouter au CHB

Les nouvelles fonctionnalités à ajouter au système CHB ont été mises en évidence lors du focus-groupe. Elles réfèrent à la possibilité d'implémenter un agent conseiller pour guider les usagers lors de la modification d'ontologie, mais aussi à la possibilité de développer une visualisation de l'évolution qui soit intégrée dans l'ontologie.

Tableau VI-17 Nouvelles fonctionnalités à ajouter au CHB

Nouvelles fonctionnalités à ajouter - CHB	Occurrence
Guidage des processus d'évolutions complexes	Focus-groupe
Visualisation des changements intégrée dans l'ontologie	Verbalisation dyade 1, 2 et 3 Focus-groupe
Variantes de changements comme outil de validation de l'évolution	Focus-groupe

Guidage des processus d'évolutions complexes. La nouvelle fonctionnalité de guidage du processus d'évolution a été proposée comme réponse à la première question du focus-groupe, à savoir : voyez-vous les éditeurs d'ontologies comme des outils multifonctionnels, permettant l'édition des changements élémentaires, mais aussi complexes, ainsi que la journalisation et la visualisation des changements, tout cela intégré dans un même éditeur ?

[61] Par exemple, du bleu pour les changements qui ajoutent des classes, du bleu foncé pour ceux qui ajoutent des propriétés, du rouge pour les changements qui effacent des classes.

Les participants ont tout d'abord mis en évidence le fait que les changements complexes traduisent mieux la pensée des utilisateurs étant donné que « dans sa tête l'usager se dit, ces deux-là il faut que je les mette ensemble, que je fasse une fusion ». Ils ont aussi affirmé qu'il n'était pas envisageable d'utiliser uniquement des changements élémentaires « parce qu'on parle de sémantique, des significations complexes et du fait que presque chaque changement a des impacts sur l'ontologie qu'on devrait prendre en compte ». Dans ce contexte, il serait alors intéressant d'intégrer dans l'éditeur un module de conseils qui, en se basant sur la trace des changements et sur un ensemble de règles d'évolution, puissent fournir aux utilisateurs des recommandations sur la manière de réaliser des changements. Ainsi, ce module, et implicitement l'éditeur d'ontologies, pourrait :

- aider les utilisateurs à définir leur intention en les questionnant sur ce qu'ils veulent faire (p.ex. 'Voulez-vous faire un effacement, une fusion, etc.?') et même peut-être en leur donnant des renseignements sur ce que représente chaque changement et l'avantage d'utiliser l'un ou l'autre (p.ex. en expliquant ce qu'est un changement de fusion et en quoi il est différent de changements qui effacent des classes et ajoutent une autre classe par la suite).

- guider les utilisateurs dans l'application de changements, particulièrement pour les changements complexes (p.ex. pour la fusion, l'éditeur demanderait à l'utilisateur de cliquer sur les classes qu'il désire fusionner, de donner ensuite un nom à la classe résultante et de spécifier sa superclasse).

- proposer des stratégies personnalisées d'évolution qui guident les utilisateurs dans les choix qu'ils auront à faire pour résoudre les impacts d'un changement sur l'ontologie (p.ex.. pour la fusion de deux classes 'A' et 'B', permettre à l'utilisateur de transférer ou effacer des sous-classes et des propriétés de 'A' et de 'B', tout en l'aidant en fonction de son choix).

Visualisation des changements intégrée dans l'ontologie. La fonctionnalité de visualisation des changements a été rediscutée lors du focus-groupe, pour répondre à la deuxième question posée : quelle visualisation vous semble plus facile à comprendre et à manipuler, une à part ou une intégrée dans l'ontologie (cf. Figure VI-6) ?

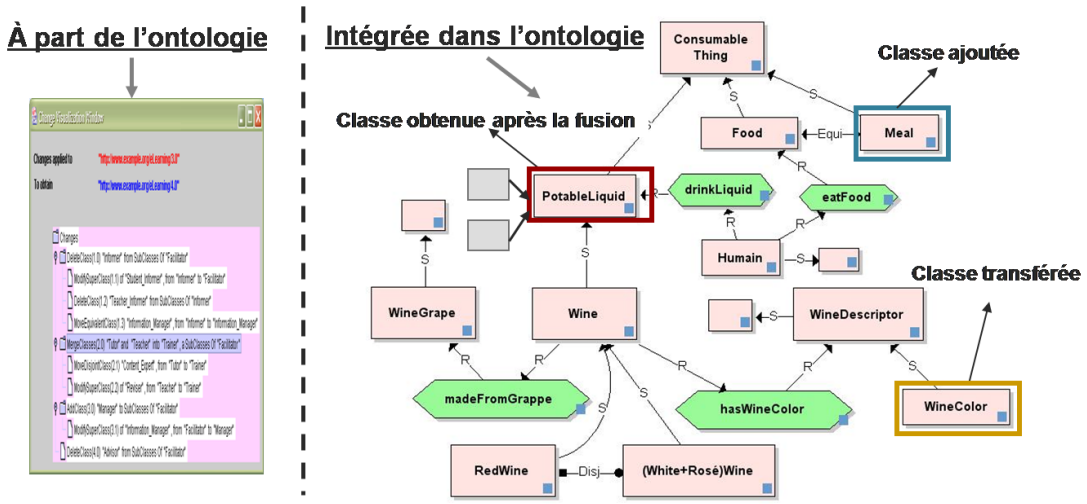


Figure VI-6. Configurations pour la visualisation des changements : (a) à part l'ontologie ; (b) intégrée dans l'ontologie

Les sujets ont précisé que les visualisations ont leur avantage : l'une permet d'avoir accès, dans une seule interface, à tous les changements ainsi qu'aux relations de causalité entre les changements, l'autre permet d'avoir un meilleur accès au contexte relié à chaque changement en particulier. Ces deux visualisations se complètent. Il ne faudrait donc pas renoncer à une au profit de l'autre, mais plutôt créer un lien entre les deux afin de permettre le passage de l'une à l'autre : « si on clique sur la classe obtenue après la fusion, on aura accès à la liste des changements, […] si on clique sur un changement dans la liste on voit le résultat dans l'ontologie ».

6.3.2. **Quelques conclusions préliminaires issues de l'évaluation du SAM**

Les conclusions préliminaires présentées dans ce paragraphe découlent d'une écoute de la verbalisation effectuée par les sujets lors des étapes 3 et 5 de l'expérimentation. Bien que, dans le cas du SAM, nous n'avons pas poussé l'analyse des données plus loin, les observations tirées nous offrent la possibilité de dégager un certain nombre de conclusions, même si celles-ci devraient probablement être validées d'une manière plus méthodique.

6.3.2.1. **Évaluation de l'analyse des changements fournie par SAM**

Nous avons regroupé les observations préliminaires en deux catégories : compréhension des effets des changements ; utilité/validité de l'analyse des changements.

6.3.2.1.1. *Compréhension et résolution des effets des changements*

Tous les participants démontrent une meilleure compréhension des effets de changements, après avoir consulté l'analyse présentée par SAM : ils donnent des réponses correctes aux exercices leur demandant de préciser, d'une manière exacte, l'effet sur l'accès direct/indirect aux ressources affectées par trois types de changements (*cf.* le Test_C, paragraphe 6.2.2.3.2.). Par contre, seulement deux sujets ont proposé des solutions exactes pour préserver l'accès direct aux ressources référencées, en affirmant qu'elles doivent être « déplacées sur la classe `ConcepteurCours` (résultante de la fusion) » et respectivement qu' « il faudrait décider vers laquelle des deux classes issues de la division il faut relier la ressource ». Les autres n'ont pu donner aucune solution valable. Ceci montre que, même s'ils sont conscients de certains effets problématiques, la plupart des sujets ont une difficulté à visualiser des solutions pour la modification du référencement sémantique des ressources, d'où l'intérêt d'avoir un système conseiller : « on a absolument besoin d'un système qui nous dit quel changement affecte quelle ressource et nous aide à résoudre les problèmes ».

Même si tous les participants ont émis un avis positif au sujet de l'analyse des changements, ils ont cependant apprécié différemment son importance. Ainsi, deux sujets ont affirmé connaître déjà une bonne partie des effets présentés, tout en estimant que cette analyse les a aidés à mieux se rappeler et comprendre certains effets. Les autres sujets ont

affirmé, par contre, qu'ils ne connaissaient que très peu d'effets et que ni ceux présentant l'accès indirect, ni ceux montrant les relations logiques entre versions d'ontologie, n'en faisaient partie. Ils ont aussi observé que, en plus du fait que certains effets « étaient moins prévisibles », ceux concernant l'accès indirect « sont vraiment importants, puisque le but de l'ontologie est aussi de permettre des inférences ».

6.3.2.1.2. *Utilité/Validité de l'analyse des changements*

Les sujets ont tous validé l'importance de l'analyse des changements, en la qualifiant d' « intéressante », « importante », mais aussi « nécessaire, dans le contexte d'un responsable d'une banque de ressources référencées sémantiquement ». Ils ont aussi affirmé que les énoncés leur semblent pertinents et corrects, tout en déplorant l'utilisation d'un vocabulaire qui demande une familiarisation préalable, mais qui est aussi trop axé sur le formalisme OWL, donc assez technique.

6.3.2.2. **Évaluation de la modification du référencement avec SAM**

Nous avons regroupé les observations préliminaires en quatre catégories : utilité de SAM ; mode assisté *vs* (semi)automatique ; pertinence des classes suggérées et choix multiples ; représentation de l'information à l'interface.

6.3.2.2.1. *Utilité de SAM – modification du référencement*

Les trois participants ont souligné l'utilité et la pertinence de la modification du référencement sémantique avec SAM, notamment dans le contexte de l'évolution des ontologies et de ressources référencées sémantiquement. Dans ce contexte, il est très difficile pour un gestionnaire des ressources d'évaluer et de résoudre les effets de changements par une modification du référencement, sans qu'il ait un système qui lui suggère des solutions possibles, qui l'aide à prendre des décisions ou même qui peut effectuer éventuellement des modifications à sa place comme c'est le cas pour les UKI affectés par des changements non problématiques.

Ces participants ont observé aussi que le fait d'avoir visualisé auparavant l'analyse de l'effet du changement (ici `MergeClasses`) les aide à mieux comprendre les suggestions de SAM et à prendre des décisions quant aux classes à choisir pour modifier les UKI affectés par le changement en question.

6.3.2.2.2. *Modification assistée vs (semi)automatique*

Les participants ont apprécié le mode conseiller de SAM pour les changements problématiques, mode nécessitant la modification de l'identifiant de la version, mais aussi celui de la classe à l'intérieur des UKI. Ils ont même précisé qu'ils n'aimeraient pas un système automatique, qui prend des décisions à leur place dans un domaine aussi sensible que le référencement sémantique. Ils préfèrent, par contre, avoir le plus de choix possibles et le contrôle sur la modification du référencement, même si cela entraîne des manipulations supplémentaires et un effort cognitif plus intense de leur part. Ces résultats ont été validés aussi lors de la séance du focus-groupe (étape 5) et ceci même avant que les participants aient interagi avec SAM.

Les participants ont également observé qu'il pourrait être intéressant de prévoir des règles prédéfinies que l'utilisateur puisse sélectionner et de paramétrer ainsi la fonctionnalité du SAM. Un exemple d'une telle règle serait la suivante : tout UKI référant aux classes fusionnées, doit être modifié pour référer, en V_{N+1}, à la classe résultante de la fusion.

6.3.2.2.3. *Pertinence des classes suggérées et choix multiples*

Les classes recommandées par le système SAM pour la modification des UKI, par exemple la classe résultant de la fusion pour le changement `MergeClasses`, ont été considérées comme étant les plus pertinentes par tous les participants. Ceux-ci ont même affirmé que cela aurait été leur choix dans la plupart de cas. Les autres classes peuvent s'avérer aussi appropriées, comme l'ont d'ailleurs souligné les participants en prenant

l'exemple de la classe `DesignerIMS_LD`[62]. On obtient ainsi un « modèle complet des modifications possibles » qui pourrait combler les attentes d'un grand nombre d'utilisateurs.

Le fait d'avoir accès à plusieurs solutions a été également validé. Les participants ont précisé que, puisque SAM ne peut pas « savoir ce qui se passe dans la tête des gens », c'est mieux d'offrir aux utilisateurs un spectre large de solutions et de lui donner la possibilité de choisir celle qui semble la plus appropriée pour la modification du référencement. De plus, observé par un des participants, le fait de « visualiser » plusieurs solutions pourrait donner aux utilisateurs des « nouvelles idées auxquelles ils n'ont jamais pensé auparavant ».

6.3.2.2.4. *Représentation de l'information à l'interface*

La manière de représenter à l'interface les suggestions proposées par SAM pour la modification des UKI a été remise en question par les participants. Ils ont tous affirmé qu'elles devraient être organisées d'une manière plus suggestive. Par exemple, présenter uniquement la classe recommandée, et ensuite, à la demande des utilisateurs, présenter les autres classes. Ou bien, même si on les représente sur une même liste, utiliser un code couleur pour différencier les classes recommandées des autres classes qui le sont moins.

En ce qui concerne les commentaires textuels associés aux classes, les participants ont trouvé qu'il est indiqué de les afficher en même temps que les classes, contrairement aux autres caractéristiques, par exemple les sous-classes ou les axiomes transférés, qui devraient plutôt être affichées à la demande pour aider, par exemple, les utilisateurs indécis à faire leur choix de modification.

Finalement, pour mettre en contexte les conseils de SAM et pour mieux les comprendre, les participants ont suggéré de donner accès, parallèlement, à la fenêtre de modification du référencement, aux versions de l'ontologie ainsi qu'à la fenêtre de visualisation des changements.

---

[62] Les participants ont pris l'exemple de la sous-classe `DesignerIMS_LD` n'ayant pas été transférée à la classe `ConcepterCours`, résultante de la fusion. Dans ce cas, il se peut qu'on ne veuille pas référer à la classe résultante de la fusion, étant donné que celle-ci n'inclut pas `DesignerIMS_LD`, mais bien à cette dernière.

6.4 **Conclusion**

Dans ce chapitre, nous avons présenté l'évaluation des systèmes CHB et SAM. Nous avons commencé par présenter les critères d'évaluation appliqués ainsi que la procédure adoptée. Ensuite, nous avons analysé les résultats d'évaluation des systèmes afin d'en tirer des conclusions concernant les deux systèmes développés dans cette thèse et qui permettront de les améliorer.

Concernant le système CHB, tous les sujets ont validé son utilité. Les principales conclusions montrent que la possibilité de visualisation des changements, offerte par le système, permet aux utilisateurs de mieux comprendre l'évolution de l'ontologie. En effet, en plus de faciliter la compréhension de l'évolution en rendant les changements explicites et en les structurant, le système permet aux utilisateurs de développer une vue correcte de ces changements. De plus, l'accès aux changements complexes, et pas uniquement aux changements élémentaires, offre aux utilisateurs la possibilité de cerner la logique d'évolution et l'intention de celui qui a effectué les modifications et de se créer une image plus correcte de l'évolution.

Cependant, l'interface du système comporte quelques lacunes. Principalement, la description trop technique des changements rend la lecture difficile. Il serait bénéfique que les changements soient décrits en langage naturel. De plus, un code de couleur pourrait être utilisé pour différencier les changements par type. Pouvoir visualiser les versions de l'ontologie en parallèle avec la liste des changements faciliterait la compréhension, comme cela a été souligné par l'ensemble des participants.

Le focus-groupe a aussi identifié des fonctionnalités qui enrichiraient le système. Principalement, les participants auraient aimé que les changements puissent être annotés (contexte, auteur, motivation…) par ceux qui les effectuent. Ceci documenterait l'évolution et en faciliterait la compréhension. Les participants ont aussi souligné l'idée de pouvoir accéder à une visualisation des changements intégrée à l'ontologie même.

Concernant SAM, les participants ont validé la pertinence de disposer d'une analyse des effets des changements ainsi que d'un outil les aidant à corriger ces effets en modifiant

les référencements affectés. Il est à noter que le système est particulièrement pertinent pour les acteurs chargés de la gestion des banques des ressources référencées. Pour les autres types d'acteurs, ce support est intéressant, mais pas nécessaire pour effectuer leurs tâches. Tous les participants ont apprécié le fait que l'outil fonctionne en mode conseiller et ils ont désapprouvé la possibilité de modifications automatiques. Les participants ont trouvé également que les classes suggérées pour la modification des référencements étaient correctes sémantiquement. Ils ont aussi apprécié le fait que plusieurs classes étaient proposées, rendant ainsi leur choix plus riche.

Comme pour CHB, les énoncés des effets des changements sont trop techniques, leur formulation en langage naturel faciliterait l'utilisation du système. Les participants ont remis aussi en question la présentation de la liste des classes proposées en suggérant d'utiliser un code de couleur pour classer celles-ci par ordre d'importance. Ils auraient aimé enfin pouvoir disposer d'un accès aux versions de l'ontologie pour mettre en contexte les conseils fournis par le système SAM.

L'évaluation des systèmes CHB et SAM a permis de montrer leur pertinence et leur utilité. Ces évaluations nous ont également offert la possibilité d'identifier les prochaines étapes pour améliorer ces systèmes, principalement l'interface graphique et l'ajout de certaines fonctionnalités, comme nous le montrons dans le chapitre des conclusions et perspective (Chapitre VII).

Chapitre VII

DISCUSSIONS ET PERSPECTIVES

Dans les chapitres précédents, nous avons abordé le sujet de l'évolution des ontologies et nous avons proposé une méthodologie pour supporter cette évolution. Nous avons également présenté les deux modules qui composent notre cadre de gestion des changements apportés aux versions d'ontologie – le système CHB et le système SAM – ainsi que les résultats de leur évaluation auprès de sujets humains.

Dans ce chapitre, qui est notamment le chapitre de conclusion, nous discutons les apports et les perspectives de la thèse. Tout d'abord, nous rappelons l'originalité de notre contribution (*cf.* Section 7.1) ainsi que la problématique et les objectifs que nous nous étions fixés (*cf.* Section 7.2). Nous présentons ensuite une synthèse de résultats obtenus, en décrivant leurs apports, leurs limites ainsi que des perspectives envisageables (*cf.* Section 7.3). Nous finirons par quelques conclusions (*cf.* Section 7.4)

7.1 **Originalité de notre contribution**

Dans cette thèse, notre avons traité le sujet de l'évolution des ontologies du Web sémantique. Nous avons notamment illustré l'idée qu'une « bonne évolution » est celle qui, en plus de préserver la consistance de l'ontologie évoluée, préserve aussi l'intégrité[63] de ce qui repose déjà sur l'ontologie. Dans notre contexte, ceci est le référencement sémantique des ressources. Cet aspect nécessite toutefois une connaissance approfondie des changements effectués et de l'intention des ontologistes ayant fait évoluer l'ontologie. Sans cette connaissance, on ne peut pas identifier correctement les effets de l'évolution sur le référencement sémantique et donc, on ne peut pas maintenir l'accès et l'interprétation des ressources, après l'évolution.

Par conséquent, autant l'identification (ou la journalisation) des changements apportés aux versions d'ontologie que la préservation de l'intégrité du référencement sont des besoins essentiels dans le contexte du Web sémantique, dont l'architecture est fondée sur le référencement sémantique des ressources, mais aussi dans le contexte des systèmes d'apprentissage en ligne, dont les composantes sont référencées sémantiquement. Cependant, à notre connaissance, très peu de travaux de recherche[64] traitent de la journalisation des changements. De plus, aucun ne prend en compte les changements complexes, ni l'utilisation d'un formalisme standard pour la représentation des changements apportés aux versions d'ontologie. Encore moins nombreux sont les travaux de recherche qui portent sur l'identification des effets de l'évolution sur les ressources référencées[65]. Les travaux existants concernent uniquement les changements élémentaires, à savoir ceux qui ajoutent ou effacent des concepts ou des propriétés. Ils ne proposent aucune solution concrète pour la résolution des effets problématiques, conduisant à une perte d'accès aux ressources, par exemple.

---

[63] Par la notion de **préservation de l'intégrité du référencement sémantique**, nous comprenons le maintien de l'accès aux ressources et une interprétation correcte de ces dernières, au moyen de la nouvelle version de l'ontologie.
[64] Comme nous l'avons montré dans la Section 4.1
[65] Comme nous l'avons montré dans la Section 5.1

Dans cette thèse, nous avons apporté des solutions novatrices pour combler ces lacunes. Nous avons proposé une approche de journalisation des changements, élémentaires et complexes, apportés à une version d'ontologie V_N pour obtenir une nouvelle version V_{N+1}. Cette approche est fondée sur un modèle uniformisé, mais aussi riche sémantiquement, car il a comme base une ontologie des changements, également développée dans cette thèse. Nous avons fourni aussi un formalisme de représentation des changements, basé sur OWL, qui rend interopérable et partageable l'historique des changements. De même, nous avons contribué d'une façon originale à l'avancement de la problématique du Web sémantique en proposant un ensemble des stratégies de modification du référencement sémantique affecté par l'évolution des ontologies. Ces stratégies sont fondées sur une analyse présentant l'impact de chaque type de changement sur l'accès aux ressources référencées, mais aussi sur leur interprétation.

Nous avons regroupé toutes ces solutions dans un cadre, intégrateur et modulaire, composé du système CHB (*ChangeHistoryBuilder*), pour la gestion de l'historique des changements, et du système SAM (*SemanticAnnotationModifier*), pour la gestion du référencement sémantique des ressources. L'originalité de ce cadre s'impose par le fait que, loin d'être dépourvu de toute intervention humaine, il s'incarne dans une société distribuée d'agents humains et logiciels qui s'échangent des informations et des services pendant, mais surtout, après l'évolution des versions d'ontologie :

- Pendant l'évolution : (a) un ontologiste modifie une version d'ontologie V_N ; (b) CHB récupère et archive les changements à l'intérieur de la nouvelle version V_{N+1}.

- Après l'évolution : (a) un acteur, humain ou logiciel (p.ex. SAM), demande à CHB d'identifier les changements apportés aux versions d'ontologie ; (b) CHB lui présente l'historique des changements, mais aussi les stratégies appliquées lors de l'évolution.

- Toujours après l'évolution : (a) un responsable des banques de ressources rend disponible le référencement sémantique, exprimé sous la forme d'un fichier des UKI qui référent à une version V_N; (b) SAM interprète ce fichier, demande au CHB d'identifier les changements effectués pour passer de V_N à V_{N+1} et fournit une analyse des effets des changements sur le référencement des ressources; (c) au

besoin, le responsable demande la modification du référencement pour le rendre compatible avec V_{N+1}; (d) SAM lui offre des suggestions, des conseils et le guide à faire des choix des modifications qui soient pertinents.

7.2 **Rappel de la problématique, des objectifs et des travaux effectués**

Les travaux que nous avons menés dans cette thèse ont comme but de répondre à un certain nombre de questionnements que nous avons identifiés comme étant cruciaux pour la gestion de l'évolution des ontologies du Web sémantique. Ces questionnements s'énonçaient comme suit. Comment supporter méthodologiquement l'évolution des ontologies ? Quels sont les types de changements qu'on peut apporter aux ontologies OWL ? Que signifie la gestion des changements et de leurs effets sur le référencement sémantique et comment peut-on la supporter ?

Pour y répondre, nous nous étions fixé trois objectifs principaux. Tout d'abord, il s'agissait de concevoir une méthodologie présentant une vue unifiée sur l'évolution des ontologies, son déroulement, ses exigences et ses besoins. Cet objectif a été concrétisé par le développement d'une ébauche de méthodologie, présentée dans le chapitre II. Deuxièmement, il s'agissait de proposer une représentation des changements pouvant s'effectuer sur des ontologies OWL et de concevoir une méthode et un outil pour la journalisation des changements d'une manière qui rende explicite toute la richesse de l'évolution. Nous avons répondu à cet objectif dans les chapitres III et IV, en présentant notre ontologie des changements ainsi que le système CHB. Troisièmement, nous avons envisagé de concevoir une méthode et un outil pour l'analyse des effets des changements sur le référencement sémantique des ressources et pour la résolution des effets problématiques par une modification du référencement. Nous avons discuté le résultat de cet objectif, c'est-à-dire le système SAM, dans le chapitre V. Rappelons aussi que des évaluations qualitatives, présentées dans le chapitre VI, nous ont permis de valider la pertinence des nouveaux concepts et l'utilité des nouveaux systèmes que nous avons développés dans cette thèse.

Soulignons également que les deux derniers objectifs se sont précisés au fur et à mesure qu'on avançait dans la compréhension du domaine et de l'évolution des ontologies du

Web sémantique. Ayant commencé à concevoir la méthodologie d'évolution des ontologies, nous nous somme rendu compte de la complexité de cette démarche. En effet, la réalisation de chaque étape de l'ébauche de la méthodologie, proposée au début de cette thèse, demandait une méthode spécifique pouvant être considérée comme un sujet à débattre en soi. Cette ébauche nous a permis toutefois de faire ressortir des besoins urgents pour la gestion des changements et de leurs effets sur le référencement sémantique, ce qui nous a amenée à identifier nos derniers objectifs de recherche.

7.3 **Discussion des résultats de recherche : apports, limites et perspectives**

Notre thèse présente un ensemble de résultats qui montrent la portée des notions et des techniques que nous avons proposées pour la gestion des changements apportés aux versions d'ontologie. Dans cette section, nous rappelons ces résultats en présentant leurs apports, leurs limites et leurs perspectives.

7.3.1. **Méthodologie d'évolution des ontologies du Web sémantique**

Le premier résultat concerne l'ébauche de la méthodologie d'évolution des ontologies que nous avons conçue pour répondre au premier objectif de recherche.

7.3.1.1. **Apports de la méthodologie d'évolution des ontologies**

Par rapport aux travaux actuels, l'apport de notre ébauche est le regroupement, dans un cadre cohérent, de plusieurs définitions concernant l'évolution des ontologies et les types de changements applicables (élémentaire ou complexe, primaire ou additionnel), mais aussi un ensemble de repères méthodologiques pour supporter l'évolution des ontologies distribuées, lors de la phase d'évolution (étapes 1 — 6) et lors de la phase de mise en opération de l'évolution (étapes 7 — 8).

7.3.1.2. **Perspectives de la méthodologie d'évolution des ontologies**

Les huit étapes du processus d'évolution sont décrites à un niveau assez général, sans présenter des détails concrets des méthodes et des principes qui pourront les supporter. Il convient aussi de préciser que cette ébauche de méthodologie n'a pas été complètement évaluée, ni par des experts, ni par l'usage, même si elle intègre certains éléments déjà utilisés dans le contexte de l'évolution des ontologies.

Pour concevoir une méthodologie complète, il serait souhaitable de faire un retour sur la méthodologie afin d'enrichir la phase de mise en opération de l'évolution en prenant en compte l'expérience que nous avons acquise en développant les systèmes CHB et SAM. Pour enrichir la phase d'évolution, il serait aussi intéressant de considérer des éléments nouveaux provenant des méthodologies de construction des ontologies, plus avancées à l'heure actuelle. En effet, nous pourrons nous intéresser au raffinement d'ontologie par une extraction des concepts, des relations et des axiomes, comme proposé par Sure, Staab et Studer, (2004), afin d'enrichir l'étape de découverte des changements de notre méthodologie. Nous aimerions aussi intégrer les principes de Gomez-Perez, (2003), pour permettre la vérification de la consistance de la nouvelle version de l'ontologie. Ou encore, pour gérer les situations conflictuelles lors de l'étape de validation de cette nouvelle version, il serait intéressant de s'inspirer du protocole de construction coopérative des ontologies, développé par Euzenat, (1995).

7.3.2. **Ontologie des changements**

Notre deuxième résultat concerne la conception d'une ontologie des changements, représentée en OWL, ayant plus de 90 concepts organisés taxonomiquement, définis par des restrictions des propriétés (7 propriétés déclarées) ainsi que par des axiomes prédéfinis du langage OWL. Cette ontologie répond, en partie, à notre deuxième objectif de recherche.

7.3.2.1. **Apports de l'ontologie des changements**

Le fait de concevoir une ontologie des changements applicables à des ontologies OWL-DL est très nouveau dans le domaine. Jusqu'à l'heure actuelle très peu des classifications sont

proposées dans la documentation scientifique du domaine et encore moins formalisées explicitement. Par rapport à ces classifications, les apports de notre ontologie des changements sont multiples. Premièrement, cette ontologie est représentée dans le langage standard OWL, ce qui la rend exploitable par tout agent informatique compatible avec ce langage. Deuxièmement, elle propose une hiérarchie des changements complexes (approximativement 40 opérations), en plus d'une hiérarchie des changements élémentaires. De plus, chaque élément de type `ComplexChange` ou `ElementaryChange` est défini explicitement au moyen des restrictions des propriétés[66] décrivant la source et la cible du changement ainsi que son objet d'application. Trois propriétés de type de données nous ont également permis d'introduire l'auteur et la possibilité d'expliciter l'ordre d'application des changements. Finalement, le troisième apport de notre ontologie consiste dans l'explicitation d'une hiérarchie des effets des changements, le `ChangeEffect`, et les liens que ces effets entretiennent avec chaque changement en particulier.

7.3.2.2. **Perspective de l'ontologie des changements**

La principale perspective de l'ontologie des changements tient à son utilisation future : nous envisageons d'orienter entièrement la démarche de journalisation des changements en fonction de cette ontologie. Bien entendu, cette dernière devrait, tout d'abord, être validée du point de vue de sa consistance, mais aussi du point de vue de la conceptualisation spécifiée.

7.3.3. **Le modèle de journalisation du système CHB et le langage OC de formalisation des changements**

Le troisième résultat concerne notre réponse au deuxième objectif de recherche. Nous avons conçu un modèle de journalisation, ainsi qu'un langage de formalisation des changements, les deux étant à la base du système CHB, que nous avons développé pour assurer la gestion de l'historique des changements.

---

[66] Ces propriétés d'objet sont : `haveSource`, `haveTarget`, `appliedOn`.

7.3.3.1. **Apports du système CHB**

7.3.3.1.1. *Au niveau du processus de journalisation*

Peu de travaux de journalisation sont proposés actuellement dans le domaine du Web sémantique. En outre, ces travaux permettent uniquement de traiter quelques changements élémentaires selon des schémas plutôt non uniformisés, de les exprimer dans des langages machine spécialisés et de les stocker dans des fichiers-journal n'ayant aucun lien avec les ontologies de référence.

Par rapport à ces travaux, les apports de notre système CHB sont multiples. Tout d'abord, il propose un modèle de métadonnées fondé sur l'ontologie des changements, permettant aux divers éditeurs d'ontologies de journaliser des changements, aussi élémentaires que complexes, d'une manière uniformisée, cohérente et complète. En raison du fait que ces métadonnées expriment une caractérisation des données d'entrées (`ArgumentX`) et de sortie (`ArgumentY`) pour chaque type de changement, elles peuvent être utilisées pour étendre les fonctionnalités des éditeurs d'ontologies actuels qui se résument uniquement à l'ajout ou à l'effacement des éléments d'une ontologie.

Dans le cadre du CHB, nous avons également développé le langage OC (*OntologyChange*) pour la formalisation standard des changements apportés aux versions d'ontologie. Ce langage, bien que fondé sur un nombre minimal de constructeurs `oc`, offre une expressivité suffisante pour représenter l'ensemble complet des changements élémentaires et complexes. Ceci, parce qu'il utilise, en plus de ses constructeurs spécifiques, les constructeurs fournis par le langage OWL pour exprimer des classes, axiomes et propriétés. De cette manière, le CHB peut fournir une trace des changements dont la signification de chaque modification est bien explicitée, tout comme la signification de stratégies d'évolution adoptées lors de l'application des changements additionnels. Tout agent capable de manipuler le formalisme OC+OWL pourrait alors interpréter cette trace. C'est ce que fait d'ailleurs le système SAM qui, en fonction des changements formalisés, peut déduire des solutions pour la modification du référencement sémantique affecté par l'évolution d'une ontologie.

CHB offre également un moyen d'éviter les problèmes d'accès aux fichiers-journal, étant donné qu'il insère la trace des changements dans la nouvelle version d'ontologie, en obtenant ainsi la version V_{N+1}Change. Pour cela, il utilise le constructeur `owl:VersionInfo`, dont le rôle est de fournir des informations à propos de l'ontologie en question. Ceci préserve la compatibilité entre la version V_{N+1}Change et OWL, mais offre, en même temps, la possibilité d'avoir accès à l'histoire des changements effectués pour obtenir cette version.

7.3.3.1.2. *Au niveau de la compréhension de l'évolution*

L'évaluation du système CHB, réalisée avec des sujets humains, nous a permis de valider l'utilité du service d'identification et de visualisation des changements. En effet, tous les sujets ont souligné la nécessité de connaître les changements apportés aux versions d'ontologie. De plus, ces sujets ont démontré une meilleure compréhension de l'évolution, après avoir consulté la liste des changements affichée à l'interface du système. Le fait d'avoir accès aux changements complexes leur a également permis de se rendre compte de la 'vraie' logique d'évolution ainsi que de l'intention des ontologistes ayant modifié l'ontologie.

7.3.3.2. **Perspectives du système CHB**

Concernant l'utilisation de l'ontologie des changements, bien que la démarche de journalisation du CHB utilise les concepts spécifiés dans la hiérarchie `ChangeOperation`, elle ne repose pas entièrement sur cette ontologie et ne peut donc pas profiter entièrement de la sémantique qui y est représentée. Par conséquent, comme perspective à court terme, nous envisageons d'orienter la démarche de journalisation en fonction de l'ontologie des changements. Ceci nécessiterait toutefois la réingénierie du système CHB pour le rendre capable de l'exploiter comme modèle de journalisation, chaque fichier-journal étant une instance précise de cette ontologie. Malgré l'effort de réingénierie, le résultat nous semble très prometteur puisqu'il nous permettra de profiter des moteurs d'inférences déjà existants, comme RACER (Haarslev et Moller, 2001), par exemple, qui est sans doute le plus connu et l'un des plus utilisés dans le domaine d'ontologies OWL. Notre système pourrait alors

effectuer des raisonnements plus complexes[67] en utilisant les informations fournies par les instances de l'ontologie des changements (qui explicitent l'historique de l'évolution), mais aussi celles exprimées dans la nouvelle version V_{N+1}Change.

Une des limites du système proposé réside dans le fait que nous n'avons pas encore fait effectivement communiquer le CHB avec un éditeur d'ontologies. Par conséquent, nous n'avons pas encore pensé à une démarche d'édition fondée sur l'utilisation de changements complexes, en plus des changements élémentaires. Bien que les changements complexes puissent mieux exprimer l'évolution des ontologies, ils demandent toutefois plus d'effort cognitif de la part des ontologistes pour choisir le changement approprié, à savoir celui qui correspondrait effectivement à leur intention. Il serait alors souhaitable de développer des méthodes de travail pour guider les ontologistes à choisir et à utiliser des changements complexes lors de l'évolution et de les implémenter informatiquement. Ceci est d'ailleurs une des perspectives de cette thèse. Pour la mettre en œuvre, nous considérons que l'outil MOT+ONTO pour l'édition graphique des ontologies OWL, en cours de développement au centre LICEF, serait le mieux adapté à nos besoins.

L'interface du système CHB peut sembler relativement sommaire du fait d'une description trop technique des changements ou encore du fait que l'exploration des changements est séparée de celle des ontologies. Pour l'améliorer, plusieurs solutions sont envisageables. Il serait souhaitable d'exprimer les changements dans un langage plus proche du langage naturel, mais aussi de les souligner éventuellement à l'aide d'un code couleur. Il serait aussi intéressant de rendre disponible un outil de visualisation des ontologies à partir de l'interface du CHB. De cette manière, les sujets pourraient consulter, en parallèle, les changements et les versions d'ontologie. Un autre aspect à prendre en compte est celui d'offrir la possibilité de visualiser les changements directement dans l'ontologie et d'avoir ainsi accès au contexte d'occurrence d'un changement.

[67] Par exemple, la trace de changements indiquera qu'une sous-classe, nommée A, a été transférée à une autre classe B et la V_{N+1}Change indiquera les propriétés de B, héritées maintenant par la classe A.

7.3.4. **L'analyse des effets des changements (le système SAM)**

Notre quatrième résultat concerne la première fonctionnalité du système SAM, c'est-à-dire celle destinée à fournir aux utilisateurs une vue sur les impacts de certains changements sur le référencement sémantique des ressources. Ce résultat correspond à notre troisième objectif de recherche.

7.3.4.1. **Apports de l'analyse des changements (le système SAM)**

7.3.4.1.1. *Au niveau de l'analyse des effets des changements*

Par rapport aux travaux actuels, qui soulignent, pour la plupart d'entre eux, uniquement les effets des changements élémentaires sur l'accès (direct) aux ressources, et cela, d'une manière plutôt abstraite, l'analyse fournie par SAM, comporte trois contributions principales.

La première concerne la possibilité de mettre en contexte cette analyse en présentant aux utilisateurs uniquement les changements (et les effets) qui touchent effectivement le référencement sémantique dont ils sont responsables. Pour cela, chaque utilisateur devrait toutefois envoyer au système SAM son propre référencement, sous la forme d'un fichier des UKI, sinon le système lui fournit une analyse générale où le *pire des scénarios* est présenté pour tous les changements apportés à V_N pour obtenir la nouvelle version V_{N+1}.

La deuxième contribution concerne le fait que l'analyse fournie par SAM prend en compte les changements élémentaires et complexes. Ceci est essentiel si on veut avoir une interprétation pertinente de la modification conceptuelle introduite dans la V_{N+1} et de son effet sur le référencement sémantique des ressources.

Finalement, la troisième contribution touche à la richesse de l'analyse qui se compose d'une : (1) identification des effets sur l'accès direct ou indirect aux ressources référencées ; (2) mise en évidence de l'inconsistance de l'interprétation de ces ressources, si besoin ; (3) découverte des relations logiques entre les classes appartenant à la version V_N et celles appartenant à la version V_{N+1}.

7.3.4.1.2. *Au niveau de l'utilité de l'analyse des changements*

L'évaluation du SAM nous a permis de tirer des conclusions concernant l'utilité de cette analyse pour des acteurs humains. Tous les acteurs ont validé l'importance de l'analyse des effets des changements en la qualifiant comme 'importante' et 'nécessaire' dans le contexte de la gestion des ressources référencées. Ils ont également témoigné d'une meilleure compréhension des effets après avoir consulté cette analyse et ils sont devenus conscients des problèmes engendrés par des changements problématiques (comme la fusion ou la division des classes, par exemple) et du fait qu'ils auront besoin d'un système qui les aide à résoudre ces problèmes.

7.3.4.2. **Perspectives de l'analyse des changements (le système SAM)**

L'analyse des changements, fournie par SAM, traite uniquement de l'effet isolé d'un changement, sans considérer aucune conjonction entre les effets produits par plusieurs changements apportés à un même élément (par exemple, le cas où on ajoute une équivalence à une classe et on lui efface une disjonction, ces deux changements ayant des effets opposés respectivement de type spécialisation et généralisation de classe). En perspective, nous aimerions enrichir cette analyse en considérant aussi les effets des changements multiples. Pour cela, une solution sera de fonder l'analyse sur l'ontologie des changements qui comprend, en plus de la spécification des changements, la spécification explicite des leurs effets.

7.3.5. **La modification du référencement sémantique (le système SAM)**

Finalement, le dernier résultat concerne la deuxième fonctionnalité du système SAM, c'est-à-dire la suggestion des modifications du référencement sémantique et un guidage des utilisateurs dans leur choix parmi ces modifications. Ce résultat correspond à notre troisième objectif de recherche.

7.3.5.1. **Apports de la modification du référencement sémantique (le système SAM)**

Avec la notion de référencement sémantique évolutif, à l'origine du système SAM, la principale contribution de ce dernier est d'avoir avancé sur un chemin, jusqu'à maintenant non exploré, bien qu'essentiel pour assurer la mise en opération des ontologies évolutives. SAM offre ainsi des solutions tout à fait nouvelles et des fonctions exploratoires pour la modification du référencement des ressources, exprimé sous la forme d'un ensemble des UKI. Le but est de préserver l'accès aux ressources et la consistance de leur interprétation au moyen de la nouvelle version d'ontologie.

Soulignons aussi deux autres contributions. Premièrement, celle d'avoir conçu le système SAM comme un 'conseiller', qui aide, qui offre des suggestions et qui guide les utilisateurs dans leur tâche, non pas comme un 'décideur' qui fait des choix à leur place. De cette manière, nous avons placé l'utilisateur au centre du processus. Qu'il s'agisse de valider la modification des UKI (le cas des changements non problématiques) ou, plus important, de choisir parmi un nombre de modifications possibles (le cas des changements problématiques), c'est l'utilisateur qui en est responsable parce que c'est lui qui sait quelle modification sera la plus appropriée à son contexte. L'aspect 'conseiller' a été très apprécié par les participants à l'évaluation du SAM, tout comme le fait d'avoir accès à un spectre large de solutions pertinentes pour la modification des UKI.

Une autre contribution de SAM concerne la pertinence sémantique des suggestions fournies pour la modification des UKI affectés par des changements problématiques. En ayant accès à l'historique et à l'analyse des effets des changements, le système SAM est capable d'offrir une palette assez complète de solutions pour rendre les UKI compatibles avec la nouvelle version d'ontologie. Un exemple concluant est celui du changement `MergeClasses`. Pour ce cas, le système SAM propose aux utilisateurs de remplacer les UKI référant aux classes fusionnées avec des UKI référant à la classe résultante, ou aux sous-classes directes de classes fusionnées, ou encore aux classes ayant reçu un certain nombre de ces sous-classes directes.

7.3.5.2. **Perspectives de la modification du référencement sémantique (le système SAM)**

Étant donné que la modification du référencement sémantique, supportée par le système SAM, est une fonctionnalité principalement exploratoire, nous ne parlerons pas ici de ses limites, mais de ses perspectives. Ainsi, tout d'abord, il faudrait envisager de compléter cette fonctionnalité en prenant en compte un spectre plus large de changements à traiter ou même des changements combinés (pour le moment SAM permet uniquement la modification des UKI affectés par des effacements, des fusions et des divisions des classes).

Il serait aussi intéressant de prévoir un ensemble de règles qui donneraient à chaque utilisateur la possibilité de paramétrer le niveau d'assistance (ou d'automatisme) du SAM. Un exemple d'une telle règle est : 'pour chaque changement `MergeClasses`, les UKI référant aux classes fusionnées doivent être modifiés pour référer, en V_{N+1}, à la classe résultante de la fusion'.

Nous pourrions également penser à un moyen pour aider les utilisateurs à définir de nouveaux UKI, non pas uniquement à modifier ceux existants. Ceci sera particulièrement approprié pour les changements qui ajoutent de nouvelles classes (ou propriétés) à l'ontologie.

Afin de faciliter l'utilisation du SAM, il faudrait aussi rendre l'interface plus conviviale et penser à organiser les informations d'une manière plus suggestive. Nous proposons l'utilisation d'un code couleur pour représenter, par ordre d'importance, les classes suggérées par SAM pour la modification des UKI. Ou encore, afficher les caractéristiques de ces classes (p.ex. les sous-classes ou les axiomes transférés) uniquement à la demande des utilisateurs. Ou même, d'intégrer à l'interface du SAM un outil de visualisation des ontologies afin de permettre aux utilisateurs de mieux comprendre leurs choix de modification.

Finalement, nous aimerions rendre SAM capable d'interpréter différents formats de représentation du référencement sémantique, en plus des UKI, afin de pouvoir le greffer sur divers outils d'annotation existants à l'heure actuelle.

7.4 **En conclusion**

En résumé, cette thèse apporte un ensemble de propositions pour la journalisation des changements apportés aux versions d'ontologie et pour la mise à jour du référencement sémantique des ressources en fonction de ces changements. Ces points ne constituent pas des idées indépendantes, mais un cadre global auquel nous nous sommes efforcée de donner une réelle cohérence. Un cadre qui répond à un réel besoin dans le contexte du Web sémantique et qui démontre la pertinence et la nouveauté des idées que nous avons développées tout au long de cette thèse. Un cadre qui nous permet de continuer les travaux selon les perspectives formulées, mais qui nous donne également la possibilité d'amorcer la construction des espaces d'apprentissage (par projet) en ligne, fondés sur le référencement sémantique de ses composantes au moyen des ontologies évolutives.

APPENDICE. A

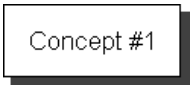
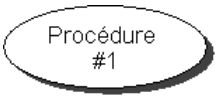
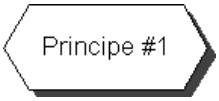
TECHNIQUE DE MODÉLISATION PAR OBJETS TYPÉS MOT

La technique MOT a été élaborée par (Paquette, 2002). Elle utilise, entre autres, trois types d'objet abstraits (concept, procédure, principe) ainsi que des liens entre ces objets (intrant/produit, spécialisation, composition, composition multiple, régulation, précédence). Dans cette thèse, nous l'avons utilisée pour la plupart des figures illustrant le fonctionnement des systèmes CHB et SAM.

A.1 Type d'objet MOT standard

Dans le modèle de type MOT standard, les connaissances abstraites sont classées en trois catégories principales (Concept, Procédure et Principe). Chacun de ces objets est représenté à l'écran par une forme graphique différente. Le tableau suivant illustre l'interprétation des types d'objet et fonction de la forme graphique lui étant associée.

Tableau A-1. Interprétation des types d'objet en fonction des formes graphiques

Type	Interprétation
Concept 	Décrit la nature des entités d'un domaine – **le quoi**. - classes d'objets ou d'événements, types de documents, catégories d'outils, catégories des personnes, …
Procédure 	Décrit des opérations permettant d'agir sur les concepts – **le comment**. - opérations génériques, tâches et activités, instructions, scénarios, …
Principe 	Décrit les contraintes, les conditions ou les acteurs d'une procédure (**le pourquoi**, **le quand** et **le qui)** - propriétés, contraintes, relations de cause à effet, lois, règles de décision, prescription, agent régulateur, acteur, … Avec le signe A : un acteur qui régit une procédure. Avec le signe O : un outil qui supporte une procédure.

A.2 Type des liens entre objets MOT standards

Les divers types d'objets peuvent être liés entre eux différents types de liens. Nous donnons un aperçu dans le tableau suivant.

Tableau A-2. Interprétation des types de liens

Type de lien	Interprétation
I/P (intrant-produit)	Met en relation : - un concept entrant à une procédure (le concept est une ressource à la procédure) - une procédure à un concept sortant (le concept est un produit de la procédure)

S (spécialisation)	Met en relation deux objets standard de même type dont l'un est « une sorte de », un cas particulier de l'autre.
C (composition)	Met en relation un type d'objet à l'une de ses composantes ou de ses parties constitutives.
C* (composition multiple)	Met en relation un type d'objet à plusieurs composantes de même type.
R (régulation)	Met en relation un principe qui conditionne un concept, une procédure ou un autre principe. Relie aussi un acteur à la procédure qu'il doit accomplir, ou un outil à la procédure qu'il doit supporter.
P (précédence)	Met en relation deux procédures ou principes dont le premier doit être terminé ou évalué avant que le deuxième ne commence.

APPENDICE. B

LE LANGAGE OWL

Le langage OWL (Antoniou et van Harmelen, 2004; Bechhofer et Goble, 2001) est dédié à définir formellement de classes et de types de propriétés, donc des ontologies. Inspiré de la logique de descriptions, il fournit un grand nombre de constructeurs permettant d'exprimer de façon très fine les classes et leurs propriétés, la contrepartie de cette expressivité est l'indécidabilité du langage obtenu. C'est pour cela qu'OWL a été fractionné en trois sous-langages d'expressivité croissante :

- *OWL-LITE*, d'une expressivité minimale et une calculabilité maximale, est prévu pour la description des hiérarchies de classes et propriétés et de contraintes simples, adéquates pour formaliser des thesaurus ou des taxonomies ;

- *OWL-DL*, d'une expressivité maximale sans perte de calculabilité (c.-à-d. toutes les inférences sont calculables et le calcul se fait en un temps fini) inclut tous les constructeurs de OWL-LITE, mais il ajoute en plus de constructeurs fondés sur la logique descriptive afin d'offrir des propriétés de calcul nécessaires aux systèmes de raisonnement ;

- *OWL-Full* permet la meilleure expressivité dans la description des ontologies, mais n'offre aucune garantie de calcul étant donné que nombre des restrictions de OWL-DL sont levées (p.ex. une classe peut être considérée en même temps comme un ensemble d'instances et comme une instance d'une autre classe).

Dans cette annexe nous allons présenter les principaux constructeurs du langage OWL, spécifiquement ceux d'OWL-DL sur lequel nous nous focalisons, accompagnés d'exemples de leur utilisation. Nous commençons par les constructeurs OWL de méta-description de l'ontologie, ensuite avec ceux reliés aux classes et propriétés, pour finir avec les constructeurs permettant de définir des instances de classes et de propriétés.

B.1 Constructeurs OWL pour la méta-description de l'ontologie

Pour prendre en compte la nature distribuée du Web sémantique, l'OWL permet d'une part d'employer des ontologies existantes pour compléter la définition d'une nouvelle

ontologie et d'autre parte de déclarer des informations sur l'ontologie elle-même. Dans le Tableau B–1, nous présentons quelques constructeurs OWL à cet effet.

Tableau B-1 Constructeurs pour la méta-description d'ontologies

<table>
<tr><th>Constructeur OWL</th><th>Expression OWL (exemple)</th></tr>
<tr><td>owl:Ontology
- permet de déclarer des méta-informations sur l'ontologie ; p.ex. rdf:comment pour des commentaires ou rdf:label pour la dénomination de l'ontologie.</td><td rowspan="5"><rdf:RDF
Déclaration d'espaces de nommage>

<owl:Ontology rdf:about="OntologieActeurs(v0.3)">
<owl:priorVersion
rdf:resource="OntologieActeurs(v0.2)"/>
<owl:incompatibleWith
rdf:resource="OntologieActeurs(v0.2)"/>
<owl:versionInfo
commentaires sur la versions/>
</owl:Ontology>

<!-- Définition de classes, propriétés et instances-->

</rdf:RDF></td></tr>
<tr><td>owl:versionInfo
- donne de renseignements sur la version de l'ontologie à l'aide des littéraux.</td></tr>
<tr><td>owl:priorVersion
- identifie l'ontologie désignée comme étant une version précédente de l'ontologie en question.</td></tr>
<tr><td>owl:backwardCompatibleWith
- indique la compatibilité de l'ontologie avec une version antérieure.</td></tr>
<tr><td>owl:incompatibleWith
- indique l'incompatibilité de l'ontologie avec une version antérieure.</td></tr>
</table>

B.2 Constructeurs OWL-DL pour la description de classes

Les classes représentent un mécanisme d'abstraction pour regrouper des entités ayant des caractéristiques similaires. À chaque classe OWL, un ensemble d'individus (c.-à-d. instances) peut lui être associé, en formant ainsi l'*extension de la classe.* Les classes OWL sont définies à travers les *descriptions de classe* et sont caractérisée à l'aide des *axiomes de classe*. Une description de classe déclare une classe OWL: (1) au moyen d'un nom de classe, (2) en énumérant les éléments qui composent une classe, (3) en utilisant des opérateurs d'intersection des classes, d'unions de classes ou l'opérateur précisant des classes complémentaires, (4) en spécifiant l'extension de classe d'une classe anonyme en utilisant

une restriction de propriété de type `owl:Restriction`. Dans le Tableau B-2, nous présentons une analyse des constructeurs de descriptions de classes en OWL-DL.

Tableau B-2. Constructeurs pour la description de classes OWL-DL

Description de classe OWL-DL	Expression OWL (quelques exemples)
`owl:Class` (s'utilise comme entête pour chaque description de classe)	

`rdf:ID`
- décrit une classe par le biais d'un nom, c.-à-d. un identifiant (ID) de classe. Cette déclaration n'a aucune sémantique particulière, si ce n'est que celle d'affirmer l'existence d'une classe ayant comme étiquette un identifiant (ID) bien précis.
	--- Déclare l'existence d'une classe, nommée `ConcepteurCours` et le fait qu'elle représente exactement un `Enseignant` qui est, en même temps, `DesignerPédagogique` et `Professeur`. ---
	`<owl:Class rdf:ID="ConcepteurCours">`
	`  <rdfs:subClassOf rdf:resource="#Enseignant"/>`
	`  <owl:intersectionOf rdf:parseType="Collection">`
	`  <owl:Class rdf:about="#DesignerPedagogique"/>`
	`  <owl:Class rdf:about="#Professeur"/>`
	`  </owl:intersectionOf>`
	`</owl:Class>`

`owl:oneOf`
- décrit une classe en énumérant exhaustivement ses instances.
	--- Défini une classe, nommée `Professeur`, par l'énumération directe de ses membres (`Pierre`, `Jean`, etc…). ---
	`<owl:Class rdf:ID="Professeur">`
	`  <owl:oneOf rdf:parseType="Collection">`
	`  <owl:Thing rdf:about="#Pierre"/>`
	`  <owl:Thing rdf:about="#Jean"/>`
	`  Etc...`
	`  </owl:oneOf>`
	`</owl:Class>`

`owl:intersectionOf`
`owl:unionOf`
`owl:complementOf`
- ces constructeurs correspondent aux opérations logiques ET (intersectionOf), OU (unionOf) et NON (complementOf).
- l'extension de la classe contient les instances communes aux classes intersectées, ou celles appartenant à au moins une des classes unifiées, ou encore les instances qui n'appartiennent pas à la classe complémentaire.
	--- Définit la classe `PersonnelUniversite` comme l'union de la classe des enseignants et de celle des étudiants.
	`<owl:Class rdf:about="#PersonnelUniversite">`
	`  <owl:unionOf rdf:parseType="Collection">`
	`  <owl:Class rdf:about="#Enseignant"/>`
	`  <owl:Class rdf:about="#Etudiant"/>`
	`  </owl:unionOf>`
	`</owl:Class>`

`owl:Restriction`
- décrit une classe anonyme qui est la classe de toutes les instances qui satisfont la restriction. Il s'applique aux propriétés pour restreindre leur valeur ou leur cardinalité (cf. section A.3).
	--- Déclare l'existence d'une classe, nommée `Doctorant` et le caractérise comme étant un `Apprenant` qui `estInscrit` aux `Cours3Cycle`. ---
	`<owl:Class rdf:ID="Doctorant">`
	`  <rdfs:subClassOf rdf:resource="#Apprenant"/>`
	`  <rdfs:subClassOf>`
	`  <owl:Restriction>`
	`  <owl:onProperty rdf:resource="#estInscrit" />`
	`  <owl:allValuesFrom rdf:resource="#Cours3Cycle"/>`
	`  </owl:Restriction>`
	`  </rdfs:subClassOf>`
	`</owl:Class>`

La déclaration d'une classe nous apprend peu au sujet de ses caractéristiques. C'est bien ici qu'interviennent les axiomes de classes, dont le but est de définir les caractéristiques nécessaires et suffisantes d'une classe (*cf.* Tableau B-3).

Tableau B-3. Constructeurs pour la déclaration des axiomes de classes OWL-DL

Axiome de classe OWL-DL	Expression OWL (quelques exemples)

`rdfs:subClassOf`
- définit une relation de sous-classement entre deux classes OWL.
	--- Déclare la classe `Enseignant`, sous-classe d'`ActeurApprentissage`, comme étant disjointe de tout `Apprenant` ---
	`<owl:Class rdf:ID="Enseignant">`
	`<rdfs:subClassOf rdf:resource="#ActeurApprentissage"/>`
	`<owl:disjointWith rdf:resource="#Apprenant"/>`
	`</owl:Class>`

`owl:equivalentClass`
- définit une relation d'équivalence entre deux classes en affirmant qu'elles ont précisément les mêmes instances. Cet axiome affirme l'équivalence extensionnelle, mais non l'identité intensionnelle (une même signification).
	--- Déclare que la classe `ProfesseurTéléuniversité` est équivalente avec la classe de toute personne qui enseigne à la Téléuniversité ---
	`<owl:Class rdf:ID="ProfesseurTéléuniversité">`
	`<owl:equivalentClass>`
	`<owl:Restriction>`
	`<owl:onProperty rdf:resource="#enseigneA" />`
	`<owl:someValuesFrom rdf:resource="#Téléuniversité"/>`
	`</owl:Restriction>`
	`</owl:equivalentClass>`
	`</owl:Class>`

`owl:disjointWith` - définit une relation de disjonction entre deux classes en affirmant qu'elles n'ont aucune instance commune.	

B.3 Constructeurs OWL-DL pour la description de propriétés

Pour déclarer des relations entre les classes, le langage OWL propose un nombre des constructeurs de propriétés de classes, que nous présentons dans le tableau ci-dessous.

Tableau B-4. Constructeurs pour la déclaration des propriétés en OWL-DL

Propriété OWL-DL	Expression OWL (exemples)

`owl:ObjectProperty`
- déclare une relation binaire entre deux classes.
`owl:DataTypeProperty`
- déclare une relation binaire entre une classe et des littéraux ou type de données XML-Schéma.
	--- Déclare une propriété d'objet, nommé `fourniAide`, dont le domaine est soit un `ConcepteurCours`, soit un `Tuteur`, et dont le codomaine est un `Apprenant`. ---
	`<owl:ObjectProperty rdf:ID="fourniAide">`
	`<rdfs:domain>`
	`<owl:Class>`
	`<owl:unionOf rdf:parseType="Collection">`
	`<owl:Class rdf:about="#ConcepteurCours"/>`
	`<owl:Class rdf:about="#Tuteur"/>`
	`</owl:unionOf>`

`rdfs:domain` - relie une propriété à une description de classe afin de spécifier son domaine	

<table>
<tr>
<td>d’application.

rdfs:range

- relie une propriété à une description de classe, ou à un littéraux ou type de donné du XML-Schéma, afin de spécifier son domaine de valeurs.

Plusieurs déclarations rdfs:domain/range sont permises et s’interprètent comme une intersection. Pour signaler cependant que plusieurs classes font office de domaine (ou de codomaine) il faudrait utiliser la structure de classe unionOf.</td>
<td rowspan="2"></owl:Class>
</rdfs:domain>
<rdfs:range rdf:resource="#Apprenant"/>
</owl:ObjectProperty>

--- Déclare une propriété de type de donnée, nommé aSigle, qui affirme que tout cours est siglé par une chaîne de caractères. ---

<owl:DatatypeProperty rdf:ID="aSigle">
<rdfs:domain rdf:resource="#Cours" />
<rdfs:range rdf:resource="etxsd; xsd:string"/>
</owl:DatatypeProperty></td>
</tr>
<tr>
<td>rdfs:subPropertyOf

- définit une relation de sous-classement entre deux propriétés d’un même type.</td>
</tr>
</table>

En plus de ces constructeurs, OWL prévoit des mécanismes supplémentaires pour construire des propriétés complexes, comme ceux pour définir de :

- relations à d’autres propriétés : owl:equivalentProperty pour déclarer l’équivalence de deux propriétés, ou owl:inverseOf pour spécifier une propriété comme l’inverse d’une autre.

- caractéristiques logiques : owl:TransitiveProperty déclare une propriété comme étant transitive et owl:SymmetricProperty la déclare comme symétrique.

- contraintes de cardinalité globale : owl:FunctionalProperty ou owl:InverseFunctionalProperty.

- contraintes de cardinalité simples : owl:Cardinality, owl:minCardinality ou owl:maxCardinality permettent de définir le nombre exact, minimal ou maximal de valeurs associées à une propriété.

APPENDICE. C

REPRESENTATION GRAPHIQUES DES ONTOLOGIES OWL-DL À L'AIDE DE MOTPLUS-ONTO

Cet appendice est tiré du manuel de l'utilisateur de MOTPlus-Onto (version 1.6.5), rédigé en 2007, par Jacques Rivard, avec la collaboration de Claire Banville, Martin Deveault, Denis Gareau, Michel Léonard, Stefan Mihaila, Gilbert Paquette et Ioan Rosca.

C.1 Type d'objet du modèle ontologique

MOTPlus permet de créer des objets graphiques représentant des types d'objets abstraits (Concept et principe), des types d'objets concrets (faits), des types de relations entre les objets précédents liens ainsi que des commentaires. Dans le modèle de type ontologique, les objets sont classés en trois catégories (Concept, Principe, Fait), et les éléments OWL-DL sont représentés chacun par un sous-type d'objet dans chacune de ces catégories (*cf.* Figure C-1).

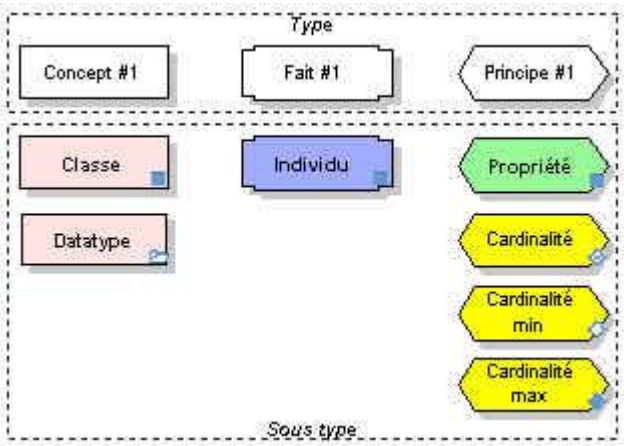
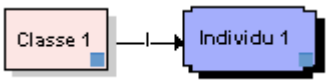


Figure C-1. Sous type d'élément OWL

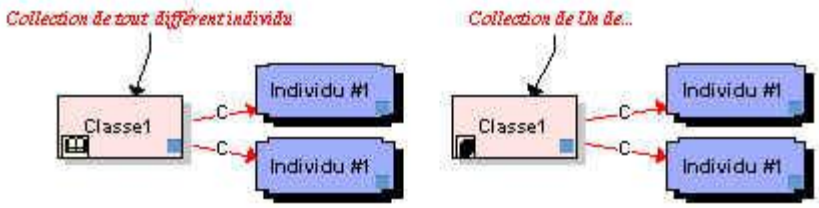
C.2 Description des liens du modèle ontologique utilisés

Dans le type de modèle Ontologie de MotPlus, les types d'objets peuvent être liés entre eux par plusieurs types de liens. Cependant, nous présentons ici seulement les que nous avons utilisés dans le document présent, l'interprétation de l'ensemble de liens disponibles étant accessible à partir du guide d'utilisation de MOTPlus-Onto (REF. Guide).

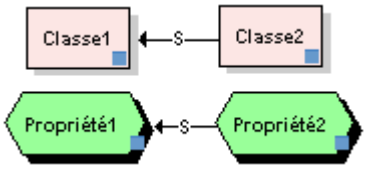
- **Le lien d'Instanciation (I)** relie une classe à un Individu (Instance de la classe).



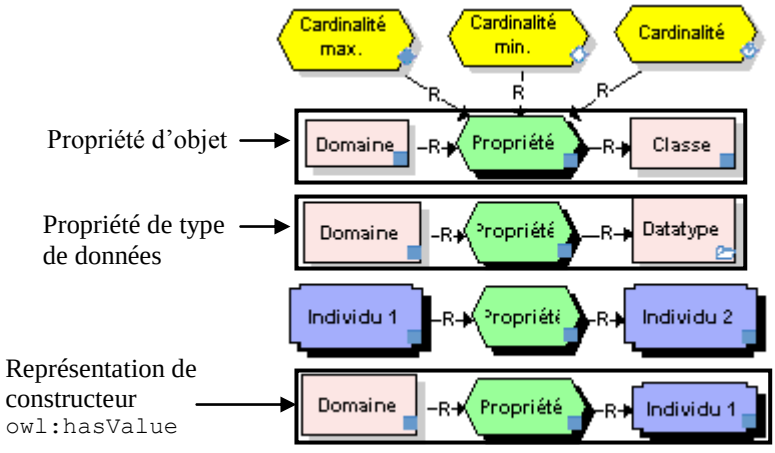
- **Le lien de Composition (C)** relie une Classe à une collection d'individus.



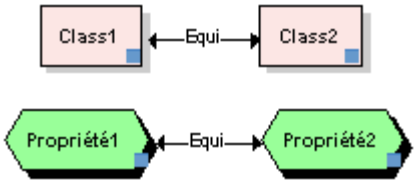
- **Le lien de Spécialisation (S)** met en relation deux Classes ou deux Propriétés dont l'une est « une sous-classe» ou « sous-propriété » de l'autre.



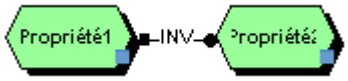
- **Le lien de Régulation (R)** met en relation, dans un sens ou dans l'autre, une propriété et une classe, un datatype ou un individu, afin d'exprimer le domaine et le codomaine de la propriété. Le lien (R) met aussi en relation les cardinalités et les propriétés, afin de créer des restrictions.



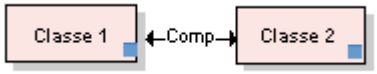
- **Lien Équivalent (Equi)** permet de relier une description de classe à une autre description de classe pour déclarer leur équivalence. Toutefois, les classes équivalentes ne sont pas égales, c'est-à-dire n'ont pas la même signification intensionnelle (ne représentent pas le même concept). Ce lien déclare aussi que deux propriétés sont équivalentes.



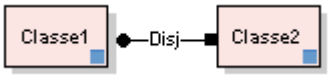
- **Lien Inverse (Inv)** déclare une propriété comme étant l'inverse d'une autre.



- **Lien Complément de (Comp**) relie une classe à son complément (c.-à.-d. la classe dont l'extension contient exactement les individus qui n'appartiennent pas à l'extension de la description de classe faisant l'objet de la déclaration).



- **Lien Disjointe (Disj)** relie deux descriptions de classe disjointes, qui n'ont aucun individu en commun.



C.3 Description des étiquettes du modèle ontologique utilisées

Les tableaux ci-dessous indiquent, pour les Objets concernés, leurs étiquettes disponibles et la description de chacune d'elle.

Tableau C-1. Étiquettes prédéfinies utilisées pour des classes

Étiquette	Symbole	Description
Union de		La description d'une classe se définit alors par les individus qui apparaissent dans au moins une des sous-classes de la collection qui s'y rattache. *Par exemple, la classe 'Caractéristique du vin' se définit par individus d'au moins une de la collection des classes 'Couleur' et 'Goût'*

Intersection de		Cette primitive décrit une classe dont l'extension de classe contient précisément les individus qui ont en communes l'extension de classe de toutes les descriptions de classe dans la liste. *Par exemple, on pourrait décrire que la classe 'Vin blanc' contient tous les individus de la collection des classes 'Bordeaux blanc' et 'Vin blanc de la Loire'.*
Tout different individu		Déclare que tous les individus de la collection de la classe sont mutuellement différents. *Par exemple, dans la collection des individus de la classe 'Enfants', tous sont mutuellement différents.*
Un de		Les membres de la collection de la classe sont exactement l'ensemble des individus énumérés, ni plus ni moins. *Par exemple, la collection d'individus membres de la classe 'JoursDeLaSemaine' correspond précisément aux 7 jours de la semaine.*

Tableau C-2. Étiquettes prédéfinies utilisées pour des propriétés

Étiquette	Symbole	Description
Symétrique		On peut déclarer des propriétés comme étant symétriques. *Par exemple, on pourrait déclarer amiDe comme étant une propriété symétrique. Un raisonneur, sachant que Frank est un ami de Deborah, pourra alors déduire que Deborah est un ami de Frank.*
Transitive		On peut déclarer des propriétés comme étant transitives. *Par exemple, si 'Sara' est une ancêtre de 'Louise' et si Louise est une ancêtre de 'Deborah' alors un raisonneur pourra en déduire que Sara est un ancêtre de 'Deborah'.*
Fonctionnelle		Contraint une propriété d'avoir une seule et unique valeur pour chaque instance. *Par exemple, la propriété aPourEpoux peut être déclarée fonctionnelle, ce qui veut dire que chaque instance appartenant au domaine 'Femme' de cette propriété ne peut pas avoir plusieurs époux, mais cela ne veut pas dire qu'une instance de la classe 'Femme' doit avoir au moins un époux.*
Fonctionnelle Inverse		Affirme qu'il ne peut y avoir deux instances distinctes x1 et x2 telles que les couples (x1,y) et (x2,y) soient tous deux des instances de P. *Par exemple, la propriété 'aLeNuméroDeSécuritéSociale' déclarée inverse fonctionnelle, indique que chaque instance du domaine 'Personne' de la propriété, ne peut avoir qu'un seul numéro de sécurité sociale (un identificateur unique)*
Toute valeur de		Cette contrainte permet de décrire la classe de tous les individus pour lesquels toutes les valeurs de la propriété concernée sont des membres de la classe de description. Il s'agit de la contrainte *allValuesFrom*.

		Par exemple, la contrainte 'Toute valeur de' pour la propriété'aPourParent' qui a des valeurs que dans la classe 'Human'. Cela décrit la classe de tous les individus ayant des parents humains.
Quelques valeurs de		Cette contrainte permet de décrire la classe de tous les individus pour lesquels au moins une valeur de la propriété concernée est un membre de la classe de description. Il s'agit de la contrainte *someValuesFrom*. *Par exemple, la contrainte 'Quelques valeurs de' pour la propriété 'aPourEnfant' qui a des valeurs que dans la classe 'Étudiant'. Cela décrit la classe de tous les individus ayant au moins un enfant étudiant.*

APPENDICE. D

FORMALISATION DES CHANGEMENTS AVEC ONTOLOGY CHANGE

D.1 Changements de classes

Les préfixes `old` et `new` sont introduits par le CHB à l'aide de deux déclarations d'entités, le but étant de faciliter l'écriture des éléments appartenant à la version V_N (…`OntologieActeur/v2`) et respectivement V_{N+1} (…`/OntologieActeur/v3`)[68]

D.1.1 Changement « AddClass ».

« AddClass » est un changement qui ajoute une classe à l'ontologie.

Metadonnée	Valeurs	Représentation du changement
ChangeNumber	Numéro	<oc:AddClass> <oc:to> <owl:Class rdf:about="etnew;#y1"> <rdfs:subClassOf> <owl:Class rdf:about="etnew;#y2"/> </rdfs:subClassOf> </owl:Class> </oc:to> <rdfs:comment>string</rdfs:comment> <oc:additionalChanges> <--! déclarer les changements additionnels --> </oc:additionalChanges> </oc:AddClass>
ChangeType	AddEntity	
ObjectOrdrer	Primaire ou Additionnel	
ChangeOperation	AddClass	
Argument(s)	X = vide	
	Y = [y1, y2] - y1 = l'ID de la classe ajoutée. - y2 = l'ID de la superclasse de la classe ajoutée.	
AdditionalChanges	Possible	
Comments	Chaîne de caractères	

D.1.2 Changement « DeleteClass »

[68] L'expression `etold;#Professeur` se développerait en `http://…OntologieActeur/v2#Professeur`

«DeleteClass» est un changement qui efface une classe d'une ontologie.

Metadonnée	Valeurs	Représentation du changement
ChangeNumber	Numéro	<oc:DeleteClass> <oc:from> <owl:Class rdf:about="etold,#x1"> <rdfs:subClassOf> <owl:Class rdf:about="etold,#x2> </rdfs:subClassOf> </owl:Class> </oc:from> <rdfs:comment>string</rdfs:comment> <oc:additionalChanges> <--! déclarer les changements additionnels --> </oc:additionalChanges> <oc:DeleteClass>
ChangeType	DeleteEntitiy	
ChangeOrder	Primaire ou Additionnel	
ChangeOperation	DeleteClass	
Argument(s)	X = [x1, x2] - x1= l'ID de la classe effacée. - x2 = l'ID de la superclasse de la classe effacée.	
	Y = vide, signifie que la classe identifiée par x1 a été effacée	
AdditionalChanges	DeleteClass, ModifySuperClass, MoveDisjointClass, MoveEquivalentClass, DeleteEquivalentClass, DeleteDisjointClass, ModifyPropertyDomain, ModifyPropertyRange, etc…	
Comments	Chaîne de caractères	

D.1.3 Changement « ModifySuperClass »

«**ModifySuperClass**» est un changement qui modifie la superclasse d'une classe dans la taxonomie. Ce changement est équivalent, syntaxiquement et sémantiquement, avec le changement qui transfère une sous-classe d'une classe à une autre.

Metadonnée	Valeurs	Représentation du changement
ChangeNumber	Numéro	<oc:ModifySuperClass> <oc:from> <owl:Class rdf:about="etold;#x1"> <rdfs:subClassOf> <owl:Class rdf:about="etold;#x2">
ChangeType	ModifyEntitiy	
ChangeOrder	Primaire ou Additionnel	
ChangeOperation	ModifySuperClass	
Argument(s)	X = [x1, x2] - x1= l'ID de la classe dont la superclasse en	

	V_N est identifiée par x2. - x2 = l'ID de la superclasse de x1 en V_N.	</rdfs:subClassOf> </owl:Class> </oc:from> <oc:to> <owl:Class rdf:about="etnew;#y1"> <rdfs:subClassOf> <owl:Class rdf:about="etnew;#y2"> </rdfs:subClassOf> </owl:Class> </oc:to> <rdfs:comment>string</rdfs:comment> </oc:ModifySuperClass>
	Y = [y1, y2] - y1 = x1. - y2 = l'ID de la superclasse de y1=x1 en V_{N+1}.	
AdditionalChanges	Non	
Comments	Chaîne de caractères	

D.1.4 Changement « AddDisjointClass »

«**AddDisjointClass**» est un changement qui ajoute un axiome de disjonction déclarant qu'une classe C_1 est disjointe avec une autre classe C_2.

Metadonnée	Valeurs	Représentation du changement
ChangeNumber	Numéro	<oc:AddDisjointClass> <oc:to> <owl:Class rdf:about="etnew;#y1">> <owl:disjointWith> <owl:Class rdf:about="etnew;#y2" »/> </owl:disjointWith> </owl:Class> </oc:to> <rdfs:comment>string</rdfs:comment> </oc:AddDisjointClass>
ChangeType	AddEntitiy	
ChangeOrder	Primaire ou Additionnel	
ChangeOperation	AddDisjointClass	
Argument(s)	X = vide	
	Y = [y1, y2] - y1 = l'ID de la classe à laquelle a été ajouté l'axiome `owl:disjointWith`. - y2 = l'ID de la classe étant déclarée disjointe avec y1.	
AdditionalChanges	Non	
Comments	Chaîne de caractères	

D.1.5 Changement « DeleteDisjointClass »

«**DeleteDisjointClass**» est un changement qui efface un axiome de disjonction déclarant qu'une classe C_1 est disjointe avec une autre classe C_2.

Metadonnée	Valeurs	Représentation du changement
ChangeNumber	Numéro	<oc:DeleteDisjointClass> <oc:from> <owl:Class rdf:about="etold;#x1"> <owl:disjointWith> <owl:Class rdf:about="etold;#x2"»/> </owl:disjointWith> </owl:Class> </oc:from> <rdfs:comment>string</rdfs:comment> <oc:DeleteDisjointClass>
ChangeType	DeleteEntitiy	
ChangeOrder	Primaire ou Additionnel	
ChangeOperation	DeleteDisjointClass	
Argument(s)	X = [x1, x2] - x1 = l'ID de la classe d'où a été effacé l'axiome `owl:disjointWith`. - x2 = l'ID de la classe étant déclarée disjointe avec x1.	
	Y = vide, l'axiome de disjonction a été effacé.	
AdditionalChanges	Non	
Comments	Chaîne de caractères	

D.1.6 Changement « MoveDisjointClass »

«**MoveDisjointClass**» est un changement qui déplace un axiome de disjonction d'une classe à une autre (tout en préservant le deuxième argument de la relation de disjonction).

Metadonnée	Valeurs	Représentation du changement
ChangeNumber	Numéro	<oc:MoveDisjointClass> <oc:from> <owl:Class rdf:about="etold;#x1"> <owl:disjointWith> <owl:Class rdf:about="etold;#x2"/> </owl:disjointWith>
ChangeType	MoveEntitiy	
ChangeOrder	Primaire ou Additionnel	
ChangeOperation	MoveEquivalentClass	
Argument(s)	X= [x1, x2] - x1= l'ID de la classe d'où a été transféré l'axiome	

	owl:disjointWith. -x2 = l'ID de la classe étant déclarée comme disjointe avec x1. Cet ID reste le même en V_{N+1}.	</owl:Class> </oc:from> <oc:to> <owl:Class rdf:about="etnew;#y1"> <owl:disjointWith> <owl:Class rdf:about="etnew;#y2"/> </owl:disjointWith> </owl:Class> </oc:to> <rdfs:commment>string</rdfs:comment> </oc:MoveDisjointClass>
	Y = [y1, y2] - y1 = l'ID de la classe au quelle a été transféré l'axiome owl:disjointWith). - y2 = x2.	
AdditionalChanges	Non	
Comments	Chaîne de caractères	

D.1.7 Changement « AddEquivalentClass »

«**AddEquivalentClass**» est un changement qui ajoute un axiome d'équivalence déclarant qu'une classe C_1 est équivalente avec une autre classe C_2.

MetaData	Valeurs	Représentation du changement
ChangeNumber	Numéro	<oc:AddEquivalentClass> <oc:to> <owl:Class rdf:about="etnew,#x1"> <owl: equivalentClass> <owl:Class rdf:about="etnew,#x2"> </owl: equivalentClass> </owl:Class> </oc:to> <rdfs:comment>string</rdfs:comment> <oc:AddEquivalentClass>
ChangeType	AddEntitiy	
ChangeOrder	Primaire ou Additionnel	
ChangeOperation	AddEquivalentClass	
Argument(s)	X=vide	
	Y = [y1, y2] - y1= l'ID de la classe au quelle a été ajouté l'axiome owl:equivalentClass. - y2 = l'ID de la classe équivalente avec y1.	
AdditionalChanges	Non	
Comments	Chaîne de caractères	

D.1.8 Changement « DeleteEquivalentClass »

«**DeleteEquivalentClass**» est un changement qui efface un axiome d'équivalence d'une classe.

MetaData	Valeurs	Représentation du changement
ChangeNumber	Numéro	<oc:DeleteEquivalentClass> <oc:from> <owl:Class rdf:about="etold,#x1"> <owl: equivalentClass> <owl:Class rdf:about="etold,#x2"> </owl: equivalentClass > </owl:Class> </oc:from> <rdfs:comment>string</rdfs:comment> <oc:DeleteEquivalentClass>
ChangeType	DeleteEntitiy	
ChangeOrder	Primaire ou Additionnel	
ChangeOperation	DeleteEquivalentClass	
Argument(s)	X = [x1, x2] - x1 = l'ID de la classe d'où a été effacé l'axiome `owl:equivalentClass`. -x2 = l'ID de la classe étant déclarée équivalente avec x1.	
	Y = vide, signifie que l'axiome d'équivalence entre x1 et x2 a été effacé.	
AdditionalChanges	Non	
Comments	Chaîne de caractères	

D.1.9 Changement « MoveEquivalentClass »

«**MoveEquivalentClass**» est un changement qui déplace un axiome d'une classe à une autre (tout en préservant le deuxième argument de la relation d'équivalence).

MetaData	Valeurs	Représentation du changement
ChangeNumber	Numéro	<oc:MoveEquivalentClass> <oc:from> <owl:Class rdf:about ="etold;#x1"> <owl:equivalentClass> <owl:Class rdf:about="etold;#x2"/> </owl:equivalentClass> </owl:Class> </oc:from> <oc:to>
ChangeType	MoveEntitiy	
ChangeOrder	Primaire ou Additionnel	
ChangeOperation	MoveEquivalentClass	
Argument(s)	X = [x1, x2] - x1= l'ID de la classe d'où a été transféré l'axiome `owl:equivalentClass`. -x2 = l'ID de la classe étant déclarée équivalente avec x1. Cet ID reste le même en V_{N+1}.	

	Y = [y1, y2]
	- y1 = l'ID de la classe au quelle a été transféré l'axiome `owl:equivalentClass`.
	- y2 = x2.
		`<owl:Class rdf:about =″etnew;#y1″>`
		`<owl:equivalentClass>`
		`<owl:Class rdf:about=″etnew;#y2″/>`
		`</owl:equivalentClass>`
		`</owl:Class>`
		`</oc:to>`
		`<rdfs:commment>string</rdfs:comment>`
		`</oc:MoveEquivalentClass>`

AdditionalChanges	Non	
Comments	Chaîne de caractères	

D.1.10 Changement « MergeClasses »

«**MergeClasess**» est un changement qui fusionne deux ou plusieurs classes dans une nouvelle classe.

MetaData	Valeurs	Représentation du changement

ChangeNumber	Numéro	`<oc:MergeClasses>`
		`<oc:from rdf:parseType=″Collection″>`
		`<owl:Class rdf:about=″etold;#x1″/>…`
		`<owl:Class rdf:about=″etold;#xi″/>…`
		`<owl:Class rdf:about=″etold;#xn″/>`
		`</oc:from>`
		`<oc:to>`
		`<owl:Class rdf:about=″etnew;#y1″>`
		`<rdfs:subClassOf>`
		`<owl:Class rdf:about=″etnew;#y2″>`
		`</rdfs:subClassOf>`
		`</owl:Class>`
		`</oc:to>`
		`<rdfs:comment>string</rdfs:comment>`
		`<oc:additionalChanges>`
		`<--! déclarer les changements additionnels -->`
		`</oc:additionalChanges>`
		`</oc:MergeClasses>`

ChangeType	MergeEntities	
ChangeOrder	Primaire ou Additionnel	
Change Operation	MergeClasses	
Argument(s)	X =[x1, …xi, …xn] - xi = ID de classe, i =1 à n	
	Y = [y1, y2] - y1 = l'ID de la classe résultant de la fusion. - y2 = l'ID de la superclasse de la classe résultante de la fusion.	
Additional Changes	ModifySuperClass, MoveDisjointClass, MoveEquivalentClass, ModifyPropertyDomain, ModifyPropertyRange, etc…	
Comments	Chaîne de caractères	

D.1.11 Changement « SplitClass »

«**SplitClasess**» est un changement qui divise une classe dans deux ou plusieurs classes nouvelles.

MetaData	Valeurs	Représentation du changement
ChangeNumber	Numéro	<oc:SplitClass> <oc:from> <owl:Class rdf:about="etold;#x1"> <rdfs:subClassOf> <owl:Class rdf:about="etold;#x2"> </rdfs:subClassOf> </owl:Class> </oc:from> <oc:to rdf:parseType="Collection"> <owl:Class rdf:about="etnew;#y1"> <rdfs:subClassOf> <owl:Class rdf:about="etnew;#z1'"> </rdfs:subClassOf> </owl:Class> ETC… <owl:Class rdf:about="etnew;#yn"> <rdfs:subClassOf> <owl:Class rdf:about="etnew;#zn'"> </rdfs:subClassOf> </owl:Class> </oc:to> <rdfs:comment>string</rdfs:comment> <oc:additionalChanges> <--! déclarer les changements additionnels --> </oc:additionalChanges> </oc:SplitClass>
ChangeType	SplitEntity	
ChangeOrder	Primaire ou Additionnel	
Change Operation	SplitClass	
Argument(s)	X = [x1, x2] - x1 = l'ID de la classe divisée. - x2 = l'ID de la superclasse de la classe divisée.	
	Y = [(yi, zi)], avec i de 1 à n - yi = l'ID de la classe résultant de la division. - zi = l'ID de sa superclasse en V_{N+1}.	
Additional Changes	ModifySuperClass, MoveDisjointClass MoveEquivalentClass, ModifyPropertyDomain, ModifyPropertyRange, Etc…	
Comments	Chaîne de caractères	

D.2 Changements de propriétés

D.2.1 Changement « AddProperty »

«**AddProperty**» est un changement qui ajoute à l'ontologie une propriété d'objet (ObjectProperty) ou de type de données (DatatypeProperty)

MetaData	Valeurs	Représentation du changement
ChangeNumber	Numéro	<oc:AddProperty> <oc:to> --- If YP2 = null,THEN --- <owl:ObjectProperty(or DatatypeProperty) rdf:about="etnew,#YP1"/> --- If YP2 ≠null, THEN --- <owl:ObjectProperty(or DatatypeProperty) rdf:about="etnew,#YP1"> <rdfs:subPropertyOf> <owl:ObjectProperty(or DatatypeProperty) rdf:about="etnew,#YP2"/> </rdfs:subPropertyOf> </owl:ObjectProperty(or DatatypeProperty)> </oc:to> <rdfs:comment>String</rdfs:comment> <oc:additionalChanges> <--! déclarer les changements additionnels --> </oc:additionalChanges> </oc:AddProperty>
ChangeType	AddEntitiy	
ChangeOrder	Primaire ou Additionnel	
ChangeOperation	AddProperty	
Argument(s)	X = vide	
	Y = [YP1, YP2]; - YP1 = l'ID de la propriété ajoutée, suivi par l'indicatif ObjectProperty ou DatatypeProperty. - YP2 = vide/ l'ID de la super propriété de la propriété ajoutée, suivi par l'indicatif ObjectProperty ou DatatypeProperty.	
AdditionalChanges	AddPropertyDomain, AddPropertyRange	
Comments	Chaîne de caractères	

D.2.2 Changement « AddPropertyDomain »

«**AddPropertyDomain**» est un changement qui ajoute une ou plusieurs classes au domaine (*domain)* d'une propriété d'objet ou de type de donnée.

MetaData	Valeurs	Représentation du changement
ChangeNumber	Numéro	<oc:AddPropertyDomain> <oc:to> <owl:ObjectProperty(or DatatypeProperty) rdf:about="etnew,#YP"> --- If domain = Single class --- <rdfs:domain> <owl:Class rdf:about="etnew,#YD"/> </rdfs:domain > --- If domain = Multiples classes --- <rdfs:domain> <owl:Class <owl:unionOf rdf:parseType="Collection"> <owl:Class rdf:about="etnew,#YD1"/> ... <owl:Class rdf:about="etnew,#YDN"/> </owl:unionOf> </owl:Class> </rdfs:domain> </owl:ObjectProperty(or DatatypeProperty)> </oc:to> <rdfs:comment>String</rdfs:comment> </oc:AddPropertyDomain>
ChangeType	AddEntitiy	
ChangeOrder	Primaire ou Additionnel	
ChangeOperation	AddPropertyDomain	
Argument(s)	X = vide	
	Y = [YP, YD1…N]; - YP = l'ID de la propriété dont le domaine a été élargi par l'ajout d'une ou plusieurs classes, suivi par l'indicatif ObjectProperty ou DatatypeProperty. - YD1… N = l'ID de chaque classe ajoutée au domaine de la propriété YP.	
AdditionalChanges	Non	
Comments	Chaîne de caractères	

D.2.3 Changement « AddPropertyRange »

«**AddPropertyRange**» est un changement qui ajoute une ou plusieurs classes au codomaine (*range*) d'une propriété d'objet. Ce changement ne s'applique pas à la propriété de type de donnée.

MetaData	Valeurs	Représentation du changement
ChangeNumber	Numéro	<oc:AddPropertyRange> <oc:to> <owl:ObjectProperty rdf:about="etnew,#YP"> --- If range = Single class --- <rdfs:range> <owl:Class rdf:about="etnew,#YR"/> </rdfs:range> --- If range = Multiples classes --- <rdfs:range> <owl:Class> <owl:unionOf rdf:parseType="Collection"> <owl:Class rdf:about="etnew,#YR1"/> ... <owl:Class rdf:about="etnew,#YRN"/> </owl:unionOf> </owl:Class> </rdfs:range> </owl:ObjectProperty> </oc:to> <rdfs:comment>String</rdfs:comment> </oc:AddPropertyRange>
ChangeType	AddEntitiy	
ChangeOrder	Primaire ou Additionnel	
ChangeOperation	AddPropertyRange	
Argument(s)	X = vide	
	Y = [YP, YR1…N]; - YP = l'ID de la propriété dont le range a été élargi par l'ajout d'une ou des nouvelles classes. - YR1… N = l'ID de chaque classe ajoutée au range de la propriété YP.	
AdditionalChanges	Non	
Comments	Chaîne de caractères	

D.2.4 Changement « DeleteProperty »

«**DeleteProperty**» est un changement qui efface une propriété d'objet (ObjectProperty) ou de type de données (DatatypeProperty).

<table>
<tr><th>MetaData</th><th>Valeurs</th><th>Représentation du changement</th></tr>
<tr><td>ChangeNumber</td><td>Numéro</td><td rowspan="8"><oc:DeleteProperty>

<oc:from>

--- If XP2 = null, THEN ---
<owl:ObjectProperty(or DatatypeProperty) rdf:about=″etnew,#XP1″/>

--- If XP2 ≠null, THEN ---
<owl:ObjectProperty(or DatatypeProperty) rdf:about=″etnew,#XP1″>
<rdfs:subPropertyOf>
<owl:ObjectProperty(or DatatypeProperty) rdf:about=″etnew,#XP2″/>
</rdfs:subPropertyOf>
</owl:ObjectProperty(or DatatypeProperty)>

</oc:from>
<rdfs:comment>String</rdfs:comment>
<oc:additionalChanges>
<--! déclarer les changements additionnels -->
</oc:additionalChanges>
</oc:DeleteProperty></td></tr>
<tr><td>ChangeType</td><td>DeleteEntitiy</td></tr>
<tr><td>ChangeOrder</td><td>Primaire ou Additionnel</td></tr>
<tr><td>ChangeOperation</td><td>DeleteProperty</td></tr>
<tr><td rowspan="2">Argument(s)</td><td>X = [XP1, XP2];
- XP1= l'ID de la propriété effacée, suivi par l'indicatif ObjectProperty ou DatatypeProperty.
- XP2 = vide/ l'ID de la super propriété de la propriété effacée, suivi par l'indicatif ObjectProperty ou DatatypeProperty.</td></tr>
<tr><td>Y = vide</td></tr>
<tr><td>AdditionalChanges</td><td>DeleteProperty, ModifySuperPropety, DeletePropertyDomain, DeletePropertyRange, ModifyPropertyDomain, ModifyPropertyRange, Etc…</td></tr>
<tr><td>Comments</td><td>Chaîne de caractères</td></tr>
</table>

D.2.5 Changement « DeletePropertyDomain »

«**DeletePropertyDomain**» est un changement qui efface une ou plusieurs classes du domaine d'une propriété d'objet ou de type de donnée.

MetaData	Valeurs	Représentation du changement
ChangeNumber	Numéro	<oc:DeletePropertyDomain> <oc:from> <owl:ObjectProperty(or DatatypeProperty) rdf:about="etold,#XP"> --- If domain = Single class --- <rdfs:domain> <owl:Class rdf:about="etold,#XD"/> </rdfs:domain > --- If domain = Multiples classes --- <rdfs:domain> <owl:Class> <owl:unionOf rdf:parseType="Collection"> <owl:Class rdf:about="etold,#XD1"/> ... <owl:Class rdf:about="etold,#XDN"/> </owl:unionOf> </owl:Class> </rdfs:domain> </owl:ObjectProperty(or DatatypeProperty)> </oc:from> <rdfs:comment>String</rdfs:comment> </oc:DeletePropertyDomain>
ChangeType	DeleteEntitiy	
ChangeOrder	Primaire ou Additionnel	
ChangeOperation	DeletePropertyDomain	
Argument(s)	X = [XP, XD1…N]; - XP = l'ID de la propriété dont le domaine a été réduit par l'effacement d'une ou plusieurs classes, suivi par l'indicatif ObjectProperty ou DatatypeProperty . - XD1… N = l'ID de chaque classe effacée du domaine de la propriété XP.	
	Y = vide	
AdditionalChanges	Non	
Comments	Chaîne de caractères	

D.2.6 Changement « DeletePropertyRange »

«**DeletePropertyRange**» est un changement qui efface une ou plusieurs classes du codomaine d'une propriété d'objet. Ce changement ne s'applique pas à la propriété de type de donnée.

MetaData	Valeurs	Représentation du changement
ChangeNumber	Numéro	<oc:DeletePropertyRange> <oc:from> <owl:ObjectProperty rdf:about="etold,#XP"> --- If range = Single class --- <rdfs:range> <owl:Class rdf:about="etold,#XR"/> </rdfs:range> --- If range = Multiples classes --- <rdfs:range> <owl:Class> <owl:unionOf rdf:parseType= »Collection »> <owl:Class rdf:about="etold,#XR1"/> ... <owl:Class rdf:about="etold,#XRN"/> </owl:unionOf> </owl:Class> </rdfs:range> </owl:ObjectProperty> </oc:from> <rdfs:comment>String</rdfs:comment> </oc:DeletePropertyRange>
ChangeType	DeleteEntitiy	
ChangeOrder	Primaire ou Additionnel	
ChangeOperation	DeletePropertyRange	
Argument(s)	X = [XP, XR1…N]; - XP = l'ID de la propriété dont le range a été réduit par l'effacement d'une ou plusieurs classes. - XR1… N = l'ID de chaque classe effacée du range de la propriété XP.	
	Y=vide	
AdditionalChanges	Non	
Comments	Chaîne de caractères	

D.2.7 Changement « ModifySuperProperty »

«**ModifySuperProperty**» est un changement qui modifie la super-propriété d'une propriété d'objet ou de type de données.

MetaData	Valeurs	Représentation du changement
ChangeNumber	Numéro	<oc:ModifySuperProperty> <oc:from> <owl:ObjectProperty(or DatatypeProperty) rdf:about="etold,#XP1"> <rdfs:subPropertyOf> <owl: ObjectProperty(or DatatypeProperty) rdf:about="etold,#XP2"/> </rdfs:subPropertyOf > </owl:ObjectProperty(or DatatypeProperty)> <oc:from> <oc:to> <owl:ObjectProperty(or DatatypeProperty) rdf:about="etnew,#YP1"> <rdfs:subPropertyOf> <owl: ObjectProperty(or DatatypeProperty) rdf:about="etnew,#YP2"/> </rdfs:subPropertyOf > </owl:ObjectProperty(or DatatypeProperty)> </oc:to> <rdfs:comment>String</rdfs:comment> </oc:ModifySuperProperty>
ChangeType	ModifyEntitiy	
ChangeOrder	Primaire ou Additionnel	
ChangeOperation	ModifySuperProperty	
Argument(s)	X = [XP1, XP2]; - XP1= l'ID de la propriété dont la super-propriété en V_N est XP2, suivi par l'indicatif ObjectProperty ou DatatypeProperty ; - XP2 = l'ID de la super propriété en V_N, suivi par l'indicatif ObjectProperty ou DatatypeProperty.	
	Y = [YP1, YP2]; - YP1 = XP1 ; - P2 = l'ID de la super propriété en V_{N+1}, suivi par l'indicatif ObjectProperty ou DatatypeProperty.	
AdditionalChanges	Non	
Comments	Chaîne de caractères	

D.2.8 Changement « ModifyPropertyDomain »

«**ModifyPropertyDomain**» est un changement qui modifie le domaine d'une propriété d'objet ou de type de donnée, en échangeant une classe faisant partie du domaine avec une ou plusieurs autres classes.

<table>
<tr><th>MetaData</th><th>Valeurs</th><th>Représentation du changement</th></tr>
<tr><td>ChangeNumber</td><td>Numéro</td><td rowspan="8">
<pre>
<oc:ModifyPropertyDomain>
  <oc:from>
    <owl:ObjectProperty(or DatatypeProperty)
rdf:about=″etold,#XP″>
      <rdfs:domain>
       <owl:Class rdf:about=″etold,#XD″/>
      </rdfs:domain >
    </owl:ObjectProperty(or DatatypeProperty)>
  <oc:from>
    <oc:to>
    <owl:ObjectProperty(or DatatypeProperty)
rdf:about=″etnew,#YP″>

--- If domain = Single class ---
      <rdfs:domain>
       <owl:Class rdf:about=″etnew,#YD″/>
      </rdfs:domain >

--- If domain = Multiples classes ---
      <rdfs:domain>
        <owl:Class>
          <owl:unionOf rdf:parseType=″Collection″>
            <owl:Class rdf:about=″etnew,#YD1″/>
            ...
            <owl:Class rdf:about=″etnew,#YDN″/>
          </owl:unionOf>
        </owl:Class>
      </rdfs:domain>

    </owl:ObjectProperty(or DatatypeProperty)>
  </oc:to>
  <rdfs:comment>String</rdfs:comment>
</oc:ModifyPropertyDomain>
</pre>
</td></tr>
<tr><td>ChangeType</td><td>ModifyEntitiy</td></tr>
<tr><td>ChangeOrder</td><td>Primaire ou Additionnel</td></tr>
<tr><td>ChangeOperation</td><td>ModifyPropertyDomain</td></tr>
<tr><td rowspan="2">Argument(s)</td><td>X = [XP, XD];
- XP = l’ID de la propriété dont le domaine a été modifié, suivi par l’indicatif ObjectProperty ou DatatypeProperty ;
- XD = l’ID de la classe appartenant au domaine, qui a été échangée avec une/d’autres autres classes.</td></tr>
<tr><td>Y = [YP, YD1…N];
- YP = XP ;
- YD1… N = l’ID de chaque classe échangée contre XD, pour modifier le domaine de la propriété YP=XP.</td></tr>
<tr><td>AdditionalChanges</td><td>Non</td></tr>
<tr><td>Comments</td><td>Chaîne de caractères</td></tr>
</table>

D.2.9 Changement « ModifyPropertyRange »

«**ModifyPropertyRange**» est un changement qui modifie le codomaine d'une propriété d'objet en échangeant une classe faisant partie du codomaine avec une ou plusieurs autres classes.

MetaData	Valeurs	Représentation du changement
ChangeNumber	Numéro	<oc:ModifyPropertyRange> <oc:from> <owl:ObjectProperty rdf:about="etold,#XP"> <rdfs:range> <owl:Class rdf:about="etold,#XR"/> </rdfs:range> </owl:ObjectProperty> <oc:from> <oc:to> <owl:ObjectProperty rdf:about="etnew,#YP"> --- If range = Single class --- <rdfs:range> <owl:Class rdf:about="etnew,#YD"/> </rdfs:range> --- If range = Multiples classes --- <rdfs:range> <owl:Class> <owl:unionOf rdf:parseType="Collection"> <owl:Class rdf:about="etnew,#YR1"/> ... <owl:Class rdf:about="etnew,#YRN"/> </owl:unionOf> </owl:Class> </rdfs:range> </owl:ObjectProperty> </oc:to> <rdfs:comment>String</rdfs:comment> </oc:ModifyPropertyRange>
ChangeType	ModifyEntitiy	
ChangeOrder	Primaire ou Additionnel	
ChangeOperation OC(x,y)	ModifyPropertyRange	
Argument(s) of OC(x,y)	X = [XP, XD]; - XP = l'ID de la propriété dont le range a été modifié. - XR = l'ID de la classe appartenant au range, qui a été échangée avec une/des autres classes.	
	Y = [YP, YR1…N]; - YP = XP ; - YR1… N = l'ID de chaque classe échangé contre XR, pour modifier le range de la propriété YP=XP.	
AdditionalChanges	Non	
Comments	Chaîne de caractères	

APPENDICE. E

PROTOCOLE DE RECHERCHE POUR *CHANGE-HISTORY BUILDER* ET *SEMANTIC_ANNOTATION MODIFIER (COMP.1)*

Pré-test (Temps de l'activité : 10 min)

PréTest_A.1 et A.2 (questions c). Remplissez le questionnaire ci-dessus avant de commencer à explorer l'outil

1. Considérez-vous comme un acteur **explorateur** de l'ontologie, c'est-à-dire toute personne qui consulte l'ontologie dans le but de s'informer sur la conceptualisation d'un domaine. Dans ces circonstances :

a. Jugez-vous nécessaire de SAVOIR si l'ontologie a été modifiée ou non ?

☐ Nécessaire ☐ Intéressant, mais pas nécessaire ☐ Pas intéressant

Bien justifiez votre réponse

b. Jugez-vous nécessaire de COMPRENDRE comment l'ontologie a été modifiée ?

☐ Nécessaire ☐ Intéressant, mais pas nécessaire ☐ Pas intéressant

Bien justifiez votre réponse

c. Jugez-vous nécessaire de CONNAITRE les effets possibles des changements appliqués aux ontologies ?

☐ Nécessaire ☐ Intéressant, mais pas nécessaire ☐ Pas intéressant

Biens justifiez votre réponse

2. Considérez-vous maintenant comme un acteur **client** de l'ontologie c'est-à-dire toute personne qui utilise l'ontologie à des fins spécifiques (p.ex. l'annotation sémantique des ressources ou la conception d'un cours dont la structure des connaissances se base sur l'ontologie). Dans ces circonstances :

a. Jugez-vous nécessaire de SAVOIR si l'ontologie a été modifiée ou non ?

☐ Nécessaire ☐ Intéressant, mais pas nécessaire ☐ Pas intéressant

Bien justifiez votre réponse

b. Jugez-vous nécessaire de COMPRENDRE comment l'ontologie a été modifiée ?

☐ Nécessaire ☐ Intéressant, mais pas nécessaire ☐ Pas intéressant

Bien justifiez votre réponse

c. Jugez-vous nécessaire de CONNAITRE les effets possibles des changements appliqués aux ontologies ?

☐ Nécessaire ☐ Intéressant, mais pas nécessaire ☐ Pas intéressant

Bien justifiez votre réponse

Test_B.1 (Temps de l'activité : 10 min)

Veuillez maintenant ouvrir le dossier « OntologieActeurs » et explorer les deux versions de l'ontologie V_N et V_{N+1} à l'aide de MOT+Onto. Répondez ensuite aux questions suivantes.

1. En visualisant UNIQUEMENT les versions V_N et V_{N+1}, pouvez-vous dire si l'ontologie a été modifiée d'une manière :

☐ **Significative** - la version V_{N+1} présente une conceptualisation différente de V_N (c.-à-d. des nombreux changements impliquant la disparition de plusieurs classes ou la modification des caractéristiques de celles existantes)

- Si vous avez coché cette case, pouvez-vous préciser les changements appliqués :

☐ Non. Précisez pourquoi

☐ Oui. Précisez les changements

☐ **Moyennement significative** - la V_{N+1} présente la même conceptualisation que celle présentée par VN, mais avec des précisions supplémentaires (c.-à-d. l'apparition des nouvelles classes ou des nouvelles caractéristiques pour les classes existantes)

- Si vous avez coché cette case, pouvez-vous préciser les changements appliqués :

☐ Non. Précisez pourquoi

☐ Oui. Précisez les changements

☐ **Non-significative** (modification des commentaires, par exemple)

☐ **Autre.** Précisez

Test B.2 (Temps de l'activité : 30 min)

Veuillez maintenant ouvrir la fenêtre « Visualisation_0.1 », minimisée en bas de votre écran. Cette fenêtre présente les changements appliqués pour passer de V_N à V_{N+1}. Après avoir exploré les changements, veuillez répondre aux questions suivantes (voir la page suivante).

CONSEIL : un guide vous est fourni pour vous aider à clarifier l'ordonnancement et les termes utilisés pour la présentation des changements. Veuillez le consulter, au besoin.

ATTENTION: ne cliquez par sur aucun des changements !
n'ouvrez pas la fenêtre « Visualisation_0.2 » !

1. En visualisant les versions V_N et V_{N+1} AINSI QUE la liste des changements pouvez-vous indiquer si l'ontologie a été modifiée d'une manière :

☐ **Significative**. Bien justifiez votre réponse

☐ **Moyennement significative.** Bien justifiez votre réponse

☐ **Non-significative.** Bien justifiez votre réponse

☐ **Autre.** Précisez

2. Précisez, dans vos propres mots, les types de changements appliqués à V_N pour obtenir V_{N+1}. Décrivez ensuite le contexte d'occurrence du changement «ModifySuperClass» appliquée à la classe 'AnimateurActivités'.

- Types de changement appliqués au V_N pour obtenir V_{N+1}
- Contexte d'occurrence du changement «ModifySuperClass» appliquée à la classe 'AnimateurActivités'

3. Décrivez, dans vos propres mots, le changement «SplitClass» appliqué à la classe 'Tuteur' ainsi que les autres changements additionnels qui en découlent (voir le guide pour une explication de la relation entre un changement primaire et additionnel).

4. Précisez les changements appliqués (ou qui impliquent) la classe 'DesignerPedagogique'. Selon vous, quel était le but de la personne ayant effectué les changements ?

Test B.3 (Temps de l'activité : 10 min)

Consultez maintenant la fenêtre « Visualisation_0.2 », minimisée en bas de votre écran, qui présente une vue différente de l'évolution pour le même couple [V_N, V_{N+1}]. Répondez ensuite à la question suivante :

5. Maintenez-vous le même avis que celui exprimé pour la question 4, en ce qui concerne les changements qui impliquent la classe 'DesignerPedagogique' ainsi que le but de la personne ayant effectué les changements ?

☐ Oui. Expliquez pourquoi

☐ Non. Décrivez alors votre nouvel avis

6. Quelle visualisation des changements vous semble **plus parlante**, en ce qui concerne la compréhension des changements qui impliquent la classe 'DesignerPedagogique', la fenêtre « Visualisation_0.1 » ou «Visualisation_0.2» ? Justifiez votre réponse.

7. Que pensez-vous de l'intérêt (l'importance) de la visualisation des changements, pour l'acteur **explorateur** de l'ontologie ? Un acteur explorateur est toute personne qui consulte l'ontologie dans le but de s'informer sur la conceptualisation d'un domaine.

8. Que pensez-vous de l'intérêt (l'importance) de la visualisation des changements, pour l'acteur **client** de l'ontologie ? Un acteur client est toute personne qui utilise l'ontologie à des fins spécifiques (p.ex. l'annotation sémantique des ressources ou la conception d'un cours dont la structure des connaissances se base sur l'ontologie)

Test C (Temps de l'activité : 20 min)

Mettez-vous maintenant dans le contexte d'une personne responsable de la maintenance d'une banque des ressources annotées sémantiquement (c.-à-d. annotées à l'aide des classes d'une ontologie).

Explorez ensuite l'analyse des changements appliqués au V_N pour obtenir V_{N+1}, fournie par le système CHB. Pour y accéder, juste cliquez sur un changement dans la fenêtre « Visualisation_0.2 » et l'analyse de celui-ci s'affiche.

Après avoir exploré l'analyse des changements appliqués pour passer de V_N à V_{N+1}, veuillez répondre aux questions suivantes (voir la page suivante).

CONSEIL : un guide vous est fourni pour vous aider à clarifier les termes prédéfinis, utilisés dans la présentation de l'analyse des changements (p.ex. accès direct/indirect, relations logiques).

ATTENTION. Pour chaque question, vous pouvez choisir une ou plusieurs réponses.

1. Après avoir consulté l'analyse des changements, quelle est la phrase qui traduit au mieux votre pensée :

☐ Je connaissais déjà tous les effets des changements présentés dans l'analyse, mais celle-ci m'a aidée à mieux me les rappeler;

☐ Je connaissais déjà certains effets de changements. Précisez lesquels

Quant aux effets que je ne connaissais pas :

☐ Ils ne m'apparaissent pas si importants. Précisez pourquoi

☐ Ils m'apparaissent assez/très importants. Précisez pourquoi

☐ Je ne connaissais pas les effets des changements présentés dans l'analyse. Le fait de les consulter m'a rendu(e) conscient(e) qu'il y a des changements qui peuvent produire des problèmes d'accès aux ressources.

☐ Autre. Précisez :

2. Après avoir consulté l'analyse du changement « MergeClasses » appliqué à la classe 'DesignerPedagogique' et 'Professeur' et en sachant qu'il existe une ressource **R1** annotée uniquement par la classe 'Professeur' :

CONSEIL : Consultez le guide pour une explication de ce qui est une requête.

a. **choisissez** l'affirmation (une ou plus) qui vous semble vraie :

☐ La ressource R1 reste accessible pour des requêtes utilisant des classes de V_{N+1}, autre que la classe 'Professeur'.

☐ La ressource R1 n'est plus accessible pour aucune requête via V_{N+1} ;

☐ La ressource R1 reste accessible pour toute requête comme auparavant en V_N.

Bien justifiez la ou les réponses choisies

b. en se basant sur le volet «Logical Relations», **spécifiez** une résolution possible pour préserver l'accès à la ressource R via V_{N+1}?

Résolution :

3. Après avoir consulté l'analyse du changement « SplitClass » appliqué à la classe 'Pédagogue' et en sachant qu'il existe une ressource **R2** annotée uniquement par la 'Pédagogue' :

a. **choisissez** l'affirmation (une ou plus) qui vous semble vraie :

☐ La ressource R2 reste accessible pour des requêtes utilisant des classes de V_{N+1}, autre que la classe 'Pédagogue'.

☐ La ressource R2 n'est plus accessible pour aucune requête via V_{N+1} ;

☐ La ressource R2 reste accessible pour toute requête comme auparavant en V_N.

Bien justifiez la ou les réponses choisies

b. en se basant sur le volet «Logical Relations», **spécifiez** une résolution possible pour préserver l'accès à la ressource R via V_{N+1}?

Résolution :

4. Après avoir consulté l'analyse du changement « DeleteEquivalentClass» qui efface l'axiome d'équivalence entre 'Apprenant' et 'PersonnelEtudiant' et en sachant qu'il existe une ressource **R3** annotée uniquement par 'PersonnelEtudiant', **choisissez** l'affirmation (une ou plus) qui vous semble vraie :

CONSEIL : Consultez le guide pour une explication de ce qui est une requête directe/indirecte.

☐ La ressource R3 reste accessible pour toute requête utilisant la classe 'PersonnelEtudiant'.

☐ La ressource R3 reste accessible pour toute requête comme auparavant en V_N ;

☐ La ressource R3 n'est plus accessible pour aucune requête via V_{N+1} ;

☐ La ressource R3 n'est plus accessible pour une requête indirecte utilisant la classe 'Apprenant' ;

Bien justifiez la ou les réponses choisies

5. Que pensez-vous de la qualité de l'analyse de changements ? La considérez-vous comme sémantiquement valide (c.-à-d. les effets énoncés, sont pertinents et corrects sémantiquement):

☐ Oui, je considère qu'elle est sémantiquement valide. Justifiez votre réponse

☐ Oui, je considère qu'elle est valide, mais trop générale. Quoi d'autres aimeriez-vous connaître plus précisément

☐ Je ne considère pas cette analyse entièrement valide. Il existe des énoncés avec lesquels je ne suis pas d'accord. Précisez lesquels et pourquoi

☐ Non, je ne l'a trouve pas valide. Justifiez votre réponse

☐ Autre. Précisez

6. Que pensez-vous de l'intérêt (l'importance) de l'analyse des changements, particulièrement si la banque des ressources annotées dont vous êtes responsable est assez (ou très) large :

☐ Je ne suis pas intéressé(e) par cette analyse de changements puisqu'elle ne m'apprend rien de ce que je ne pouvais pas savoir juste en regardant la fenêtre de « visualisation des changements ».

☐ Je ne suis pas intéressé(e) par cette analyse puisque je ne me préoccupe pas de vérifier s'il existe des changements qui affectent mes ressources. C'est à ceux ayant fait évoluée l'ontologie de s'en préoccuper.

☐ Je suis intéressé(e) par cette analyse parce qu'elle met en évidence les changements plus problématiques et m'aide à comprendre leurs effets sur l'annotation des ressources.

☐ Cette analyse est très pertinente. Cependant, j'aimerais aussi avoir un système qui me montrera effectivement quels changements affectent quelles ressources et qui m'aiderait à en résoudre les problèmes, si nécessaire.

☐ Autre. Précisez

APPENDICE. F

EXEMPLES DES EXTRAITS DE VERBALISATION CODÉS

Tableau F-1 Exemple des extraits codés à l'aide de codes de la catégorie «connaître et comprendre l'évolution»

Code	Exemples d'extraits codés
CoEv.Sa$^{Ex, Cl}$(+);	«… c'est nécessaire (de savoir), puisqu'un domaine évolue, donc une ontologie évolue, d'où cette nécessité. […] en plus, si on est dans le contexte de l'annotation des ressources, c'est nécessaire de savoir puisqu'on fait des liens entre nos ressources et l'ontologie. […] C'est possible alors que, si l'ontologie a changé, il y en a certains liens qui ne sont plus valides.» *- Dyade formée par P2 et P6, PréTest_A.1*
CoEv.Con^{Cl}(+)	« Mais oui, par exemple je suis un concepteur et je veux utiliser une ontologie … on a deux aspects pour nécessaire (de connaître les changements appliqués, donc l'évolution). Savoir si on est toujours d'accord avec la vue du domaine. […] et aussi savoir si le cours et les applications qui en dépendent sont toujours pertinents par rapport au domaine. » *- Dyade formée par P2 et P6, le Prétest_A.1* «…peut-être qu'ils (les changements) sont valides du point de vue sémantique, mais non du point de vue du concepteur (utilisateur)…ça peut même défaire notre projet de conception du cours, donc c'est nécessaire de connaître ce qui a été fait pour pouvoir décider si on est d'accord avec l'ontologie ou non.» *- Dyade formée par P4 et P5, le PréTest_A.1*
CoEv.VueG^{S} (incompléte, incertaine, complexe et non-facilitée)	«…on ne peut pas tout identifier (les changements), on peut peut-être juste dire qu'il y a certains propriétés et axiomes qui ont changé…où non, ils ont été ajoutés puisqu'il y a de nouvelles classes.» *- Dyade formée par P4 et P5, le Test_B.1* « … il y on a des changements! […] Il y on a sûrement des changements que je ne voie pas. » *- Dyade formée par P1 et P3, le Test_B.1*
CoEv.Chg^{S}(fusion, n°3.3)_(-)^{C}	« La logique de tout ça a changé, c'est comme si Pédagogue a changé en ConcepteurCours, Informateur et Tuteur.» *- Dyade formée par P4 et P5, le Test_B.1*
CoEv.Chg^{S}(ajout,	« - PersonnelEnseignant … PersonnelEnseignant. Ça, c'est OK. Ensuite c'est Informateur et Acteur dans $V_{N+1.}$

n°2.1)_(+)^{E}(-)^{C};	- C'était l'autre qui a été ajoutée (le sujet montre la classe Informateur)» *- Dyade formée par P1 et P3, le Test_B.1*
CoEv.Chg^{S}(remplacement, n°1.2)_(+/-)^{C} **CoEv.Conf^{S}**_(complexité des changements apportés)	«Là (en V_N), le Pédagogue est un animateur d'activités. Et là (en V_{N+1}), c'est le Tuteur qui fait ça. Donc, elle a changé Pédagogue pour Tuteur … mais qu'est-ce que c'est l'Informateur ?… Ça devient compliqué, tout cela. » *- Dyade formée par P2 et P6, le Test_B.1*
CoEv.Conf^{S}_(complexité des changements apportés)	«- Et là (en V_N) c'est Pédagogue et ici (en V_{N+1}) c'est Tuteur. - Oui, Tuteur, un nouveau concept. Puis c'est le Gestionnaire …enfin, c'est un peu difficile de comprendre la logique de tout ça.» *- Dyade formée par P4 et P5, le Test_B.1*
CoEv.VueG^{A}(compléte, certaine, facilitée) **CoEv.Clarif^{A}**_(liste des changements générée par CHB) **FONCT_U**(identification et visualisation des changements)	«Du coût ça (le CHB) donne tous que … tous les changements apportés. […] En fait, c'est pas mal parce que ça (la liste des changements) aide à bien voir les changements…on n'a plus à chercher les changements sur le graphique (versions de l'ontologie en MOT+Onto) … ou on sait où chercher. » *- Dyade formée par P1 et P3, le Test_B.2* «… il y a des changements qu'on n'a pas vue comma ça, à l'œil … on ne les a pas tous vus, sans cela (la liste de changements générée par le CHB). […] Là, c'est facile d'aller voir les changements. T'imagines... juste avec MOT+ (l'éditeur d'ontologie), tu passes d'une version à une autre pour les comparer! C'est n'est pas systématique ... la, avec la liste, c'est systématique. En plus, ça aide de voir les changements.» *- Dyade formée par P2 et P6, Test_B.2*
CoEv.Comph(primaire/add, n°3.1)_(liste changements) **CoEv.Comph**(modification domaine, n°2.8)_(liste changements)	« - *Split* Pédagogue en Tuteur et Informateur … et ça a eu des conséquences qui ont modifié toutes ses propriétés. - Puis ça a effacé cela (le sujet montre, sur la liste, l'effacement de DesignerPedagogique et ses conséquences).» *- Dyade formée par P4 et P5, le Test_B.2* « - […] *from* Pédagogue à Tuteur. Ça veut dire que cela (la propriété 'encadreEtudiants') est déplacé de Pédagogue à Tuteur. - Oui, c'est ça. C'est exactement ça. C'est d'un déplacement du domaine ici qu'on parle.» *- Dyade formée par P1 et P3, le Test_B.2*

Tableau F-2. Exemples des extraits codés à l'aide de codes associés à l'évaluation du CHB : catégorie «formes de visualisation des changements»

Code	Exemples d'extraits codés
Vis.Chg.Onto(+) **Vis.Pr/Add.Onto(+)**	Comme toutes les autres dyades, les sujets P4 et P5 explorent, en tout temps, la liste des changements en faisant des allers-retours entre celle-ci et les versions de l'ontologie. *- Extrait des notes, prises durant l'expérimentation, sur le comportement non verbal des dyades*
Vis.Chg.Comp(-)_(OWL+, sugg -)	«C'est juste la façon dans laquelle c'est écrit … *add* Designer *to subclasses of* … en fait, c'est ça. C'est le Designer_IMS_LD qui est une sous-classe du Personnel et non que le Designer et la superclasse de Personnel. » *- Dyade formée par P1 et P3, Test_B.2* « - Ensuite c'est *addClass* ConcepteurCour *to PropertyDomain* … Ah, ça, je ne comprend pas, trop difficile. - Ça, c'est une propriété […], mais j'ai comme même du mal à comprendre. - C'est la façon dans laquelle c'est écrit. » *- Dyade formée par P2 et P6, Test_B.2*
Vis.Chg.Comp(+)_Onto(+)	« - […] c'est *AddDisjointClass* Étudiant Collège au EtudiantUniversitaire. - C'est ça quoi (le sujet P3 montre le lien de disjonction directement sur la version papier de V_{N+1}) […] c'est une classe disjointe. C'est la disjonction. - OK. C'est add disjonction. C'est qu'ils ont rajouté ici (le sujet P1 encercle l'axiome sur la version V_{N+1}) le lien de disjonction.» *- Dyade formée par P1 et P3, Test_B.2*

Tableau F-3. Exemples des extraits codés à l'aide de codes associés à l'évaluation du CHB : catégorie «changements complexes *versus* élémentaires»

Codes	Exemples d'extraits codés
Chg.IL_E	« - Là, la classe Designer IMS_LD elle a été ajoutée. Tu vois, elle était là (le sujet montre son emplacement sur la V_N)… elle n'a pas été déplacée, mais rajoutée […] carrément effacée et rajoutée. - Deleter puis créer une nouvelle …c'est bizarre ça.» *- Dyade formée par P2 et P6, Test_B.2* « - […] et Designer_IMS_LD est supprimée aussi. - Pourtant, le designer IMS_LD n'est pas supprimé, il est juste déplacé (le sujet montre sa confusion). […] Bon, ça été effacée puis rajoutée, enfin, tout ces actions-là (le bloc de changements additionnels, causés par l'effacement de la classe DesignerPedagogique) ont été effacées puis rajoutées, tant que dans d'autres cas (le cas de la division de la classe Pédagogue) ils ont simplement déplacé des liens en laissant les objets sur place.» *- Dyade formée par P1 et P3, Test_B.2*
Chg.CL_C	«- Le changement «ModifySUperClass» est le changement […] qui modifie la superclasse d'une classe dans la taxonomie. […] Ah ... d'accord. - […] je dirais qu'on comprend vraiment mieux les changements qui ont été faits... c'est intéressant cette vue (la fenêtre VisualisationChangements_2).» *- Dyade formée par P2 et P6, Test_B.3*
Chg.CL_C **Chg.Fac(+)_C**	«Mais…pourquoi le faire d'une telle manière (l'effacement et l'ajout de la classe Designer_Pédagogique, dans la fenêtre VisualisationChangements_1)? Pourquoi pas comme ci-dessus, pour le changement Split (changement représenté directement sous sa forme complexe), comme ici, où on transfère les sous-classes? » *- Dyade formée par P1 et P3, Test_B.2*
Chg.FCo$^{N°3.1}$	«Il y a aussi …le split de Professeur (fausse conception, en vérité la classe Professeur n'a pas été divisée, mais fusionné avec DesignerPedagogique) qui a entraîné plusieurs changements additionnels […] il y a eu aussi la modification du Pédagogue (en fait, la division est ici) …» *- Dyade formée par P4 et P5, Test_B.2*

Chg.FCo$^{N°2.1}$ **Chg.ReFCo**$^{N°2.1}$	«Le DesignerPedagogiue a été éliminé pour replacer ses deux sous-classes …» *- Dyade formée par P1 et P3, Test_B.2* «Mais en fait, elle (DesignerPedagogique) est fusionnée avec Prof. […] ce sont ces deux classes la qui sont fusionnées... avec seulement Designer_IMS_LD qui est déplacée.» *- Dyade formée par P1 et P3, Test_B.3*
Chg.But(-)_E(effacement + ajout) **Chg.But(+)_C(fusion)**	«- Moi, je dirais que la c'est difficile de savoir le but (l'intention formelle reliée aux changements appliqués à la classe DesignerPedagoqie, dans la fenêtre VisualisationChangements_1)…c'est pas évident de comprendre le rationnel qui est derrière tout ça … - Oui, c'est sur qu'on peut juste faire des hypothèses spéculatives.» *- Dyade formée par P2 et P6, Test_B.2* «Là, dans la deuxième fenêtre (fenêtre VisualisationChangements_2) […] c'est une nouvelle action, c'est une fusion. Ah, OK … c'est ça qui nous donne la raison!» *- Dyade formée par P2 et P6, Test_B.3*
Chg.Vis(+)_C **Chg.Vis (-)_E** **Chg.Fac(_)_E**	«La deuxième (fenêtre VisualisationChangements_2) […] ici c'est plus parlant parce que c'est plus condensé… les changements sont regroupés sous une seule signification.» «Dans la première […], on est obligé de laisser des choses en suspense, parce qu'il y a d'autres changements qui suivent plus bas. Donc, c'est comme s'ils sont traités en deux temps […] ce qui rend la compréhension assez difficile. » *- Dyade formée par P1 et P3, Test_B3*

APPENDICE. G

TRANSCRIPTION ET CODAGE DES VERBALISATIONS

Cet appendice présente la transcription et l'analyse des verbalisations de la dyade 1 (étape 2 de l'évaluation du système CHB). Nous avons analysé les verbalisations de deux autres dyades de manière similaire.

Tableau G-1. Transcription de la verbalisation pour l'étape 2 : dyade 1

Codes associés	Verbalisation de la dyade 1 (participants P2 et P6)
	Légende (n.a.) - note de l'auteur : clarification des notions, hypothèse, observation [] - description du comportement non verbal, si besoin EE - explication de l'expérimentatrice concernant le déroulement de l'évaluation
Pré-test A.1 (questions a et b) et A.2 (questions c)	
	Question 1.a
CoEv.Con^{Ex}(+)	P2 : En pensant à l'exemple qu'elle (l'expérimentatrice) a présenté tout à l'heure, s'il s'agit d'un domaine assez récent, quand tu y vas tous les six mois, par exemple, tu veux savoir comment ça évolue. Ou encore, si l'ontologie est réalisée d'une manière collaborative, tu ne l'as pas vue depuis un moment, tu veux savoir ce qui a été fait.
FONCT_E(annotation de changements – acteurs) CoEv.Con^{Ex}(+)	P6: Qui a modifiée… c'est aussi un aspect important, surtout dans le domaine des sciences… savoir si la modification a été faite par un certain étudiant, ou par une sommité dans le domaine. Donc, je dirais que oui… c'est nécessaire de savoir les changements, mais aussi par qui ont été apportés.
FONCT_E(annotation des changements - justification)	P2 : Attend, par la suite, c'est écrit comprendre comment l'ontologie a été modifiée… donc comprendre comment, on pourrait dire par qui aussi. S'il y a eu des discussions, s'il y a eu plusieurs personnes qui l'ont modifiée.
CoEv.Sa$^{Ex, Cl}$(+)	P2: Mais la c'est juste de savoir si l'ontologie a évolué… mois je dirais que oui, c'est nécessaire, puisqu'un domaine évolue, donc une ontologie évolue… d'où cette nécessité.
	P6 : Moi, je te dis, j'aimerais savoir la logique d'abord, pourquoi le domaine évolue, dans quel sens, quels sont les effets…
CoEv.Con^{Cl}(+)_Arg(liens entre ontologies évolutives et ressources annotées)	P2 : En plus, si on est dans le contexte de l'annotation des ressources… donc c'est nécessaire de savoir si l'ontologie a évolué… on pourrait dire une deuxième raison ce que nous… on a fait des liens entre nos ressources et l'ontologie.
	P2 : Si on est un prof de cours, en a fait notre cours au départ, chacun des matériels pédagogiques est relié à des connaissances, comme dans MISA. Là, c'est possible que, si l'ontologie a changé, il y en ait certains liens qui ne sont plus valides.
	P6 : Oui, c'est ça... On aurait besoin d'une mise à jour, pour modifier peut-être la ressource
	P2 : Vérifier la pertinence des ressources. Donc, c'est nécessaire pour vérifier la cohérence des liens entre ressources et connaissances.
	Question 1.b
CoEvCon^{Ex,Cl}(+) CoEv.VueG^{A}(complète,	P6 : C'est intéressant de comprendre l'évolution de l'ontologie, ou comment l'ontologie a été modifiée, puisque ça nous permet de comprendre l'évolution du domaine.

facilitée)	P6 : Puis, ça me permet de voir deux visualisations différentes… de comprendre l'évolution du domaine. C'est comme j'ai la une trace de deux ou trois... comme des snapshot des ontologies d'un domaine quelconque... voir les différences entre une et l'autre .. ou ça m'aide à voir comment ça a été modifiée ... comprendre, ça c'est un peu fort.
FONCT_E(annotation changements)_(identification acteur) FONCT_E(annotation changements)_(justification changement)	P2 : Donc, on pourrait mettre le 2ème commentaire que… comprendre c'est aussi répondre aux questions de pourquoi et qui (pourquoi le changement a été fait, et par qui a-t-il été fait).
	P6 : Quand j'introduis un changement, il faut que j'écrive… et que justifie pourquoi je le modifie, en tant qu'éditeur.
	P2 : Tu veux dire qu'on devrait ajouter des informations…
	P6 : Peut-être même des annotations de changements.
	P2 : … à de différentes versions, comme la date qui l'a modifiée et d'avoir un genre de commentaire pour chaque modification… ou même la discussion entre ceux qui l'ont modifiée
	P6 : À mon avis, c'est très important.
	P6 :… l'autre (le CHB) c'est comme un formaliseur, il ne comprend pas le processus… il voit comme deux moments… mais qu'est-ce qu'a fait qu'une chose évolue dans l'autre.
	Question 2.a
CoEv.Sa^{Cl}(+)	P2 : .Ah, un acteur client. On a été allés trop loin, là… [le sujet J montre la question 1.c]
	P6 : Là, pour savoir si l'ontologie a été modifiée, je dirais oui… c'est nécessaire. On l'a déjà justifiée à la question 1.a
	Question 2.b
CoEv.Con^{Cl}(+)_Arg(donner son accord avec la vue du domaine)	P6 : Mais oui… par exemple, je suis un concepteur et que je veux utiliser une ontologie, mais je suis plutôt d'accord avec une ontologie, avant sa modification qui a été faite par quelqu'un avec lequel je ne partage pas vraiment le même point de vue.
	P2 :… donc, là ça sera de comprendre comme elle (l'ontologie) a été modifiée pour voir si on est toujours en accord avec l'ontologie, qui est toujours une interprétation du domaine… mais on pourrait aussi voir si le cours (dont la structure dépend de l'ontologie, par exemple) est toujours pertinent.
CoEv.Con^{Cl}(+)_Arg(décider de la pertinence des applications basées sur l'ontologie)	P2 : Imagine… on avait « la terre est plate » dans l'ontologie. Et il y a quelqu'un qui vient dire… non, la terre est ronde… ça veut dire que ton cours d'auparavant n'est plus pertinent.
	P2 : Ce qui fait qu'on a deux aspects… pour nécessaire (n.a. nécessaire de comprendre comment l'ontologie a évolué). Savoir si on est toujours d'accord avec la «vue» du domaine. La conceptualisation du domaine, mais c'est vraie qu'habituellement, une

	ontologie est une « vue partagée » d'un domaine. Donc, si tu arrives comme prof et tu dis... ça c'est nul, tu t'es met en peu en marge dans la communauté. Mais, c'est vrai que des fois cela arrive. Il y a des profs qui voient les choses en peu différentes.
	P2 : Là, si tu veux, on a défini deux façons de voir l'ontologie. Une subjective, par rapport à des points de vue, entre lesquels il y a une relativité et par rapport auxquels on peut se positionner. Et une autre, plus objective, le domaine représenté objectivement et qu'on est obligé de respecter.
	P2 : Et aussi savoir si le cours et les applications qui en dépendent sont toujours pertinents par rapport au domaine.
	Question 2.c
	P2 et P6 : Dans ce contexte, de client, donc de l'explorateur, c'est nécessaire de connaître les effets de changements. L'explication on l'a donnée auparavant (n.a. 1.c.2)
Test B.1	
	[les sujets ouvrent à l'écran, dans MOT+, les versions V_N et V_{N+1} de l'ontologie, et commencent à l'explorer à l'écran, mais aussi dans le format papier]
	Exploration de versions d'ontologie - début (temps d'exploration - 15 minutes)
CoEv.Chg^S(ajout, n° 1.1)_(-)^E(-)^C CoEv.Conf^S_(complexité ontologie)	P2 : Là, je vois qu'il y en a des changements ici… [Le sujet montre une partie de la V_{N+1}, format papier]. Là, il y a ChercheurEnseignant (n.a dans V_N). Il est là (n.a. dans V_{N+1}), qui est une sorte de PersonnelEnseignant. Ensuite, il y a GestionnaireInscription qui a la propriété gèreInscriptionCours....ah... on l'a pas ici (n.a. en V_N)...voila un premier changement.
CoEvVueG^S(complexe, non-facilitée)	P6 : Ensuite Pédagogue … la il y en a des changements!
CoEv.Chg^S(remplacement, n°1.2)_(-)^E(-)^C CoEv.Conf^S_(complexité de changements appliqués)	P2 : C'est comme s'ils ont revu les rôles des acteurs. La il y a Pédagogue … et non, en VN+1 il y en a plus Pédagogue. Il y a, par contre Informateur, Tuteur, Concepteur de cours, Designer_IMS_LD …
	P6 : Là (n.a. en V_N), le Pédagogue est un évaluateur de connaissances et un animateur d'activités. Et là (n.a. en V_{N+1}), c'est le Tuteur qui fait ça. Donc, elle a changé Pédagogue pour Tuteur, mais qu'est-ce que c'est l'Iinformateur … ça devient compliqué, tout cela.
CoEv.Chg^S(effacement, n°1.3)_(+)^E(-)^C CoEv.Chg^S(effacement, n°1.4)_(+)^E	P2 : Et le Professeur, tu as vu… il est plus ici, dans la nouvelle version. Et t'a vu, Designer_IMS-LD était une sous-classe, et là elle devient une classe à parte entière. Donc, là il y a eu des changements [le sujet montre la partie Professeur, DesignerPedagogique, sur la V_N]… puis là il y a eu une relation d'équivalence, qu'on retrouve plus dans la V_{N+1}.
	P6 : Où? Ah, là, Apprenant est équivalent au PersonnelEtudiant.
Observation 1.1 : les changements additifs et soustractifs sont plus facilement observables.	
CoEv.ReQ(chg.	P2 :… est-ce que le reste a changé (n.a. de l'apprenant) ? Non… ah, tiens Le

n°1.1)_(exploration V_{N+1})	Gestionnaire d'inscription est ici maintenant. Il y a comme un changement, il y a la gestion des inscriptions qui a été faite par un Gestionnaire d'inscription, qui était un Pédagogue, qui était un PersonnelEnseiganant. Et puis finalement, le Gestionnaire d'inscription n'est plus la ... il est la ...
Observation 1.2 : plus les ontologies sont complexes, plus cette remise en question est difficilement envisageable	
	Exploration des versions d'ontologie - exemple fin [après l'exploration, les sujets commencent à répondre à la question du test B.1]
CoEv.Sign^{S}(significative)_ Arg(supression des classes)	P2 :… Significative, moi je dirai significative … puisque la il y a eu suppression des classes, Pédagogue surtout .. et Professeur. Puis Designer est aussi changé ici … designer IMS_LD…MISA
CoEv.Chg^{S}(remplacement, n°1.5)_(+/-)^{C} mais incomplet et incertain	P6 : Mais le Professeur n'a pas été juste supprimé, il a été modifié aussi…
	P2 :… c'est comme si les rôles du PersonnelEnseignant ont changé.
	P6 :. Et le DesignerPédagogique .. il est où …
	P2 : … il est devenu ConcepteurCours, je pense…
	P6 :… et le Designer_IMS_LD est mis ici ….
Observation 1.3 : les sujets ont bien identifié une partie de ce changement, mais il faudrait tenir compte du fait qu'ils ont passé plus de 15 minutes à explorer une petite partie de l'ontologie. Dans le contexte des ontologies « normales », cela devient extrêmement fastidieux.	
Test B.2	
	[Après avoir pris connaissance des consignes décrites dans le protocole qui leur a été fourni, les sujets commencent à explorer les changements présentés dans la fenêtre de visualisation du système CHB]
	Exploration de la liste des changements présentés à l'interface du système CHB - début
CoEv.Chg^{A}(ajout, n°1.6) CoEv.Comph(chg n°1.6)_(liste du CHB) Vis.Chg.Onto(+)	P6 : Change … add disjointClass ÉtudiantCollege au EtudiantUniversitaire …
	P2 : C'est là, ici [le sujet montre les classes sur les versions papier]. Oui, ils ont ajouté un axiome là…
Vis.Chg.Comp(-) _Arg(OWL+, sugg +/-) FONCT_E(guide de lecture)	P6: C'est quoi disjoint …
	P2 : Ça veut dire que les deux sont … enfin, c'est l'une ou l'autre, ça ne peut pas être les deux.
	P6 : OK… C'est correct ça.
Observation 1.4 : le langage ciblé OWL fait en sorte que toute personne non experte peut avoir une compréhension plus difficile de la description des changements. Il faudrait, d'une parte, décrire les changements dans un langage moins formaliste, et d'une autre parte, de créer un guide de lecture.	

Vis.Chg.Onto(+) FONCT_N(accès, en parallèle, aux versions d'ontologies et à la liste des changements) CoEv.Chg^{A}(effacement, n°1.7 ; division, n°1.8 ; effacement, n°1.9)	P2 : Ensuite, c'est DeleteEquivalentClass Apprenant *from* PersonnelEtudiant … on a vu ça... C'est celui-là [le sujet le montre sur la V_{N+1} en papier]
	P2 : Le troisième c'est le Split …
	P6 : Oui…c'est ça, *split* Pédagogue en Tuteur et Informateur …c'est là [le sujet le montre sur la V_{N+1} en papier]. … ensuite *delete* DesignerPedaogique *from* subclasses of PersonnelEnseignant.
	P2 :… c'est celui-là… [le sujet le montre sur la V_{N+1} en papier]
Observation 1.5 : les deux participants explorent TOUJOURS la liste en faisant des allers-retours entre la liste des changements, affichée à l'écran, et les versions de l'ontologie. Pour faciliter la lecture des changements, il faudrait alors rendre disponibles, en parallèle, les versions de l'ontologie.	
CoEv.VueG^{S}(incomplete, incertaine) CoEv.Chg^{A}(effacement, n°1.10 ; ajout, n°1.11 ; ajout, n°1.12)	P6 : Ensuite c'est *delete* Professeur …
	P2:. oui, le Professeur qui était la …
	P6 :… ensuite *add* classe Doctorant … ah, on n'a pas vu ça.
	P6: Add class Designer_IMS_LD to PersonnelEnseignant
Chg.IL_E(effacement et ajout de la classe correspondante au changement n°1.12) Chg.FCo$^{1.1}$(classe ajoutée et non pas déplacée)	P2 :… juste une minute. Là, la classe Designer IMS_LD elle l'a ajoutée .Tu vois, elle était la [le sujet montre son emplacement sur la V_N papier]… elle ne l'a pas déplacée, elle l'a rajoutée. Elle l'a carrément effacée et rajoutée.
	P6 :… deleter puis créer une nouvelle …c'est bizarre ça.
Vis.Chg.Comp(-)_Arg(OWL+, sugg-)	P6 : Ensuite...Add_class_ConcepteurCour_to_PropertyDomain … Ah, ça… je ne comprend pas… trop difficile.
	P2 : Ça c'est une propriété… elle est où cela? [les sujets cherchent pour un bon moment à comprendre la description]
Observation 1.6 : Pour une personne moins experte en OWL, la description de changements des propriétés s'avère très difficile puisqu'elle est trop technique, donc peu significative pour la personne en question.	
Vis.Chg.Comp(+)_Onto(+)	P2 : Ah, je comprends… ConcepteurCours la… *to* ça .Ok, la il y un lien [le sujet montre le lien R qui part de la classe vers la propriété]. OK…c'est cette classe qui est entre les deux … [Le sujet consulte le guide fourni]. Oui, c'est une propriété d'objet.
Vis.Chg.Comp(-)_(OWL+, sugg-) FONCT_E(guide OWL)	P2 : Mais j'ai comme même du mal à comprendre.
	P6 :… c'est la façon dans laquelle c'est écrit, trop technique.
	P2 : C'est juste … regarde … quand tu as une propriété comme ça … tu as un domaine et un co-domaine. Tu as, par exemple, l'Enseignant transmetteConnaissances à l'Apprenant. Mais là, tu as vu, il n'y a pas de codomaine … ce n'est pas cohérent.

Observation 1.7 : on peut identifier deux aspects pour cette incompréhension. Premièrement, il faudrait un guide OWL pour expliquer que, ontologiquement, une propriété n'est doit pas obligatoirement avoir un domaine et un codomaine. Deuxièmement, dans les versions présentées, la propriété possède un codomaine, c'est juste que le nom de la classe n'est pas présenté par manque d'espace, c'est qui peut être un déficit du protocole de l'évaluation.	
	Exploration de la liste des changements présentés à l'interface du système CHB – fin [après l'exploration, les sujets commencent à répondre à la question du test B.2]
	Question 1
CoEv.VueG^{A}(complète, certaine, facilitée) FONCT_U(liste des changements du CHB) CoEv.Sign^{A}(significative)_ (chg. complexes)	P6 : Significative, parce qu'il y a au des changements complexes … sinon, c'est comme l'autre question.
	P2 : Enfin, des changements de tout type … et, est qu'on pourrait dire qu'il y a des changements qu'on n'a pas vue comme ça, à l'œil…? On ne les a pas tous vus, sans cela (n.a. la liste).
Vis.Chg.Comp(+/-)_Arg(sugg-) FONCT_E(liste des changements + code couleur pour les mettre en évidence)	P6 :… oui, peut-être ….c'est sur qu'on aurait dû avoir une interface plus parlante, pour mieux voir (n.a. comprendre) les changements… c'est lente d'identifier touts les petits changements…. mais, bon, on n'évalue pas l'interface.
	P2 : Oui, c'est sûr que des couleurs, ça aurait aidé de passer de l'une à l'autre…[le sujet montre la liste et ensuite la V_{N+1}]
CoEv.Clarif^{A}_(liste des changements du CHB) FONCT_U(visualisation des changements)	P2 : Mais cela nous a aidé, ça, de voir la liste comme ça… la, c'est facile d'aller voir les changements. T'imagines... juste avec MOT+ (n.a. l'éditeur d'ontologie), tu passes de l'une à l'autre pour les comparer (n.a. les versions) ! C'est n'est pas systématique ... la (n.a. avec la liste), c'est systématique. En plus, ça aide de voir les changements ... et tu dois plus te demander s'ils sont ou non corrects.
	Question 2
Chg.IL_E Chg.But(-)_E Chg.Fac(-)_E E = effacement+ajout d'une même classe	P2 :… en on met en premier, suppression de classes.
	P6 :…puis ajouter de classe …
	P2 :… ajouter de classes, oui, mais cela correspond… par exemple, ici, c'est marqué Add Designer IMS_LD, mais lui il été déjà… c'est comme un déplacement dans la hiérarchie.
Observation 1.8 : il s'agit ici d'un changement complexe - ModifySuperClasse - qui a été exprimé sous la forme d'une suite des changements élémentaires - DeleteClass et AddClass. On voit très bien que, même si le sujet comprend cela d'une manière intuitive, il aurait eu besoin de visualiser du début ces changements complexes. Cela lui aurait sauvé du temps, lui aurait donné une compréhension plus correcte, sans qu'il soit obligé de faire des déductions qui peuvent s'avérer correctes ou NON.	
	P6 : Là, il y a ajout des propriétés…
Observation 1.9 : ajout du domaine de la propriété, pas de la propriété. Mais l'expression AddPropertyDomain peut facilement porter à cette confusion pour une personne non experte d'OWL, en plus d'être une expression assez technique, donc peu significative pour les sujets. Cela rejoint l'observation faite pour la dyade 2.	

CoEv.Comph(chg n°1.8) Vis.Chg.Onto(+)	P2 : Il y a le *split* aussi. Comme on pourrait appeler ça … oui, découpage d'une classe en deux… ainsi que l'ajout et la suppression des axiomes de disjonction et équivalence entre les classes, voila-ils sont la… [le sujet montre les axiomes sur la V_{N+1}]
CoEv.Comph(ModifySuperClass)_(liste du CHB) Chg.CL_C Chg.But(+)_C C = modification de la superclasse de la classe Animateur	P6 : Contexte d'occurrence d'Animateur Activité ?
	P2 : Oui...pourquoi est-il apparu (n.a. le changement)
	P2 : … le changement «ModifySuperClass» est le changement [le sujet consulte le guide fourni pour l'explication de termes utilisés à l'interface] … c'est un changement qui modifie la superclasse d'une classe dans la taxonomie. Ce changement est équivalent, syntaxiquement et sémantiquement, avec le changement qui transfère une sous-classe d'une classe à une autre. Ah... d'accord.
	P6 : Animateur était la… il était à Pédagogue et maintenant il est à Tuteur.
	P2 : On a changé sa classe supérieure … de la classe Pédagogue à la classe Tuteur, parce que Pédagogue a été splitté.
	Question 3
CoEv.Comph(chg n°1.8)_(liste du CHB) mais SANS les changements additionnels	P6 : Donc, il a splitté ici… c'est Pedagogue qui a été changé pour Tuteur et Informateur.
Observation 1.10 : Les sujets ont décrit le changement appliqué au AnimateurActivité, mais pas son contexte d'occurrence qui pourrait être identifié en se basant sur la hiérarchie des changements primaires-additionnels, explicitée dans la liste. Soit les sujets n'ont pas ouvert la hiérarchie des changements, ce qui met en évidence un problème d'accès, soit ils ne l'ont pas comprise, ce qui met en évidence un problème d'interface et de description peu suggestive.	
	Question 4
	P6 :… premièrement, ce qu'on peut observer ce qu'un Concepteur ne doit pas être nécessairement un Designer_IMS_LD …
Vis.Chg.Onto(+) Chg.Fac(-)_E Chg.FCo[1.2](+ cherche raison subjective)	P2 :…tu as vu ça? Professeur identifie connaissances et il y a le DesignerPédagogique qui fait les activités… et la [le sujet montre VN+1], le Concepteur de cours fait les activités et les connaissances
	P6 : Oui …
	P2 :…C'est comme ci, la (n.a. en VN) il y a avait comme des personnes… et la (n.a. en VN+1) c'est plus de rôles. Genre, le Prof peut être Informateur, Tuteur, Concepteur… et le Concepteur de cours peut utiliser MISA. Mais c'est ça que je ne comprends pas…IMS_LD c'est plus pour la diffusion.
Observation 1.11 : En se basant sur la comparaison entre V_N et V_{N+1}, le sujet P2 identifie intuitivement une des caractéristiques de la fusion des classes Professeur et DesignerPédagogiques, tel que présenté dans la fenêtre de visualisation VisualisationChangements_0.2. Il se questionne cependant au sujet de la véracité de son propos.	

Chg.But(-)_E(effacement +ajout de classes)	P2 : Moi, je dirais que la, justement, c'est difficile de savoir le but…c'est pas évident de comprendre le rationnel qui est derrière tout ça… juste quand on a deux photographies …
	P6 : Oui, c'est sur qu'on peut juste faire des hypothèses spéculatives.
Observation 1.12 : les sujets essaient de comprendre la raison du changement du point de vue de pourquoi il a été fait, donc le but SUBJECTIF. Ils ne tentent pas à identifier, voire comprendre, le but OBJECTIF du changement, c'est-à-dire les changements qui ont été effectivement appliqués.	
FONCT_E(annotation changements)_(justification changement)	(n.a. pour une compréhension HUMAINE, fondée sur la signification subjective, il est nécessaire de montrer, à part les changements réels appliqués - genre changements complexes *vs* élémentaires, une description de la parte de la personne ayant appliqué le ou les changements pour expliciter non seulement l'intention mais aussi celle subjective, à savoir pourquoi un changement a été appliqué.
Test B.3	
	Question 5
Chg.But(+)_C Chg.ReFCo[1.2]_C Chg.Fac(+)_C ; C = fusion des classes et changements additionnels	P2: La, dans la deuxième fenêtre, addClass Doctorant on l'a vu, la deuxième, la, c'est une nouvelle action …c'est fusion … Ah.. OK… C'est ça qui nous donne la raison!
	P2 : Ça veut dire qu'il y a eu une fusion de Professeur et DesignerPedagogique … là, ça aide. C'est nettement mieux ça, c'est comme au niveau supérieur, c'est plus élémentaire
Chg.Vis(+)_C	P6 :… C'est vrai que je vois mieux ce qui a été fait, mais est-ce que je comprends pourquoi…c'est une autre histoire.
Chg.CL_C ; FONCT_E(annotation changements)_(justification changement)	P6: Donc, je dirais qu'on comprend vraiment mieux les changements qui ont été faits, mais.. Il nous manque encore le Pourquoi … «le meaning that is in the world» … dans le social. Mais c'est intéressant cette vue, cependant.
Observation 1.13 : toujours la raison subjective, soulignée par le sujet P6 qui cherche, avant tout ce qui se passe dans la tête des personnes, et non pas leurs comportements observables. Du point de vue machine, cette pensée interne est loin de pouvoir être «comprise» sans l'aide de l'humain. Et comme notre contexte et celui du Web sémantique et des ontologies formelles, le fait de saisir cette pensée interne, ne fait pas encore partie de nos tâches.	
	Question 6
Chg.Vis(+)_C Chg.Fac(+)_C	P2 : La deuxième…
	P6 : Oui, la deuxième… en fait, je pense qu'on peut cliquer sur ça… ah, oui, regarde, ce sont les changements additionnels. Retourne sur la première fenêtre… ça, on ne l'a pas fait nous. [les sujets se sont rendu compte que maintenant du fait qu'ils n'ont pas regardé les changements additionnels]
	P2 : Clique sur Split .. Ok, on n'a pas regardé ça. Ça (n.a. les changements additionnels

	reliés à la division) c'est comme de sous-opérations.
	P2 : Mais, t'es d'accord cependant que cela (n.a. la deuxième fenêtre de visualisation) est meilleure
	P6 : Oui… c'est plus synthétique.
	Question 7
FONCT_U(identification et visualisation des changements) FONCT_E(description des changements)	P6 : L'importance de la visualisation des changements… elle est cruciale. La prise de conscience des changements doit être cependant rapide et facile
	P6 : Sans un effort cognitif…
	P2 : Tu sais, genre, quelqu'un disait, je vois une ontologie à un moment et après six il revient et veut savoir quelle est l'évolution du domaine… soit que cette prise de conscience soit rapide (n.a. critère utilisabilité).
	Question 8
	P2 : C'est pareil.

APPENDICE. H

TRANSCRIPTION DU FOCUS-GROUPE

Tableau H-1. Transcription de la discussion de groupe lors de l'étape 4

Focus-Groupe (20 min)	
Participants : Tous les six participants (P1-P6) et l'expérimentatrice qui est identifiée par E	
Éditeur multifonctionnel et intégré pour la conception et l'évolution des ontologies **Question :** Est-ce que vous voyez les éditeurs d'ontologies comme des outils multifonctionnels qui regroupent des fonctionnalités de visualisation des changements, la possibilité d'effectuer des changements plus complexes comme la fusion ou la division, des fonctionnalités d'analyse des changements, mais tout cela intégrées dans l'éditeur?	
Avantage intégration - conseils montrant comment faire l'évolution d'une ontologie grâce à l'historique et à l'analyse des changements	P1. C'est ça…c'est que si tu commences à faire des modifications, l'avantage d'avoir un tel outil est qu'éventuellement, tu peux mettre un système conseillé dedans … en disant fait attention, si tu fais ça, ça implique telle ou telle chose, si tu fais telle autre chose, ça implique autre chose. Donc, le fait que ça soit dans le même outil... tu peut avoir la trace des changements qui se font et, en même temps, tu as des conseils sur comment faire les changements.
	P3. Et aussi, même des fois quand tu commences à modifier un fichier, toi-même de fois tu ne sais plus trop... il y a deux semaines, mais d'où je suis parti déjà, mais qu'est-ce que j'ai déjà fait comme changements. Et que là, pouvoir accéder à l'historique, ça peut être très intéressant.
	P4. Mais si ce sont des changements complexes, c'est peut-être difficile à suivre par l'outil et par l'utilisateur.
	E. Mais, par exemple, pour toi comme concepteur/conceptrice, si tu as un outil qui te donne la possibilité que d'effacer ou d'ajouter et un autre outil qui te donne, en plus, la possibilité de faire de changements plus complexes. Tu sais, tu cliques sur une fonction et tu fais la fusion de A et B, par exemple. Quel outil sera plus pertinent pour exprimer ta pensée?
FONCT N(système conseiller/de guidage) - aider à clarifier l'intention; - proposer des étapes; - proposer des solutions d'évolution (pour les chg. additionnels, p.ex.).	P1. C'est aussi parce qu'on peut simplifier la manière d'appliquer les changements. On peut dire : qu'est-ce que tu veux faire? Est-ce que c'est un Merge, un Add, un Delete? Puis, une fois que tu as défini qu'est-ce que tu veux faire, le système te propose des étapes. Dans ce sens là, même si les changements sont complexes, t'es guidé par le système, il te piste sur ce qui est à faire. Je prépare, par exemple, un Merge de deux classes, le système me demande comment tu la nommes (n.a. la classe résultante de la fusion), de quelle classe à quelle classe que tu mets ensemble, comment tu la renommes, puis qu'est-ce que ça implique. Tous les changements qui sont impliqués par le Merge, tous ces changements additionnels apparaissent ...et le système te demande: qu'est-ce que tu fais avec ça, qu'est fait tu avec telle propriété ou telle sous-classe.
	E. Et cela tu peux le faire tant pour les changements complexes, mais aussi pour ceux élémentaires, par exemple, pour DeleteClass et ses changements élémentaires.
	P1. Et le système, en tenant compte de la logique de choses, peut proposer des solutions possibles pour l'évolution : tu veux effacer les sous-classes ou mieux les déplacer, pareille pour les propriétés.
Utilisabilité de	E. Oui, c'est exactement cela que je comprends par intégré. Par contre, du point

changements complexes vs élémentaires - les chg. complexes traduisent mieux la pensée de ceux qui veulent modifier l'onto.	de vue concepteur, cela peut être plus difficile à utiliser, au moins au début, qu'un outil qui te permette d'effacer et ajouter à ta guise.
	P3. Par contre, dans sa tête, je ne pense pas que le concepteur se dit j'efface ces deux classes-la, j'en fais une nouvelle …dans sa tête je pense qu'il se dit, ces deux-la il fait que je le mets ensemble, que je fasse une fusion. Pour le Split je pense que c'est pareil, il ne se dit pas en fait je vais supprimer cette classe et plus tard de créer cela, cela et cela. Dans nos têtes, on dit plutôt… mais non, cette classe la, elle doit être divisée dans d'autre à cause du telle ou telle chose.
Pertinence de l'outil intégré - la sémantique complexe demande des chg. complexes et a des effets complexes.	P1. Et pour en revenir à l'idée de l'outil simple versus l'outil intégré. Un outil simple, dans lequel tu peut juste ajouter ou enlever n'as aucun sens, parce que ça (n.a. les changements, le processus d'évolution) a toujours un impact, toujours, toujours… Quelque parte, que tu ne peux pas avoir juste une affaire de j'enlève, j'ajoute, j'enlève, j'ajoute. Ceci parce qu'on parle de sémantique, des significations complexes et presque chaque changement a des effets à prendre en compte.
Visualisation des changements à part ou intégrée dans l'ontologie À votre avis, qu'est-ce que serait-il plus intéressant… une visualisation à part de l'ontologie? Ou d'avoir cette présentation des changements intégrée dedans l'ontologie où les changements sont représentés directement dans l'ontologie ?	
FONCT_N (liste séparée et chg. intégrés) Les deux types de visualisations – séparées, mais interreliées - ex. on clic sur un aspect dans une visualisation et on a une image du même aspect dans l'autre	P1, P3 et P6: Les deux.
	P5. Oui, la possibilité d'avoir les deux, même par exemple sous la forme des pop up … et de pouvoir en choisir.
	P3. C'est-à-dire que c'est bien d'avoir les deux, mais si dans le schéma (n.a. l'ontologie représentée graphiquement en MOT+) si on click par exemple sur la classe obtenue après la fusion, la classe qui est en rouge, la on aurait les lignes de l'autre type de visualisation (n.a. l'arbre des changements additionnels, par exemple). Enfin, de pouvoir passer de l'une à l'autre. Donc, avoir les deux séparés, mais comme même d'avoir un lien entre les deux.
	P4. Ou comme dans l'éditeur de scénarios (n.a. l'outil développé au LICEF) qui est basé sur la même idée, où si on click sur un principe puis dans la hiérarchie d'à côté on voit le principe et à quoi il est lié… c'est comme une fenêtre dans ton interface et si tu click la tu le vois dans ton schéma et l'inverse.
	E. Mais, c'est l'idée, comme vous l'avez d'ailleurs tous validée, d'avoir les deux en même temps. Mois j'avais pensé soit à l'une, soit à l'autre..parce que de fois, même graphiquement on peut se perdre ou de seulement avoir la liste à côté on peut se perdre pareille. Donc l'idée c'est de le relier.
FONCT_U (visualisation intégrée) Nécessité d'avoir, à part la liste, la visualisation des chg. intégrés (ou des versions avec des indicatifs visuels sur	P3. Oui, parce que les deux se complètent réciproquement.
	P1. D'ailleurs, en regardant le modèle (n.a. l'ontologie en MOT+) on a vu des changements, en regardant la première interface on a vu d'autres changements et puis avec la deuxième interface on a vu encore qu'il y avait un Merge ici, et c'est encore plus parlant. En ce sens la, on ayant les deux, le séparé et l'intégré, c'est l'idéal. Et de toute façon, quand on regardait l'interface que tu nous avais préparée, je revenais souvent au modèle, pour identifier où ce qu'il se trouve tel ou tel changement, pour voir la structure.

les chg. appliqués)	P5. Oui, cela on l'a fait aussi. On retournait souvent à l'ontologie.
FONCT_N (variantes comme outil de chg.) - formes pour visualiser les chg. FONCT_U (variantes) - voir chg. après et pendant l'évolution; - valider l'évolution.	P6. Il y a le moyen d'enrichir l'outil de façon à voir les changements sous forme des variantes (n.a. variante de MOT+)
	E. Et cela peut être utile après l'évolution, mais aussi pendant l'évolution, si on veut vraiment voir la vue générale du processus d'évolution.
	P1. Oui, c'est important de travailler par des couches… enfin, des variantes … puisque comparer les deux de fois c'est assez lourd… et en fonction de ces variantes on peut valider ensuite l'évolution ou on peut travailler par couches pour ne pas trop modifier, revenir, modifier encore... etc..
FONCT_N(génération de ceux 2 vis. après et pendant l'évolution) FONCT_U (*idem*.) - aide à raisonner sur l'évolution ; - facilite l'évolution coopérative de l'onto.	P1. Mais ça aide aussi à raisonner, si je suis en train de modifier ça m'aide à raisonner le fait de voir les deux vues, les trois vues
	E. Alors, si je comprends bien, la visualisation des changements, sous leur forme graphique et\ou liste, on doit le générer après l'évolution, mais aussi pendant l'évolution de manière à nous aider a mieux comprendre ce que, nous ou d'autres, on fait hier, avant-hier, ce qu'on est en train de faire maintenant et ce qu'on devra faire demain.
	P4. Ceci serait surement très utile pour ceux qui travaillent ensemble sur la même chose, sur la même ontologie. Le fait de voir les changements, nous aide à comprendre ce que les autres ont déjà fait ou de ne pas se perdre après que les autres ont fait des modifications.
FONCT_E (annotation des changements par leur valeur sémantique exprimant l'intention subjective de l'acteur)	P6. J'aimerais aussi attirer l'attention sur un aspect qui me semble important pour la compréhension de l'évolution. C'est bien qu'on a deux ou trois versions de l'ontologie, qu'on peut le comparer à l'aide de la trace des changements, mais je ne pense pas que c'est assez pour bien comprendre. Parce que, c'est sur que je peux hypothetiser à partir de cela, je peux dire que, voila, la il comprend le rôle d'Enseignant comme Tuteur ou Informateur, ça, je peux le dire. Mais si il y a quelque chose dans l'outil, comme un genre de commentaire, qui me justifie pourquoi on a fait tel un tel changement, conceptuellement parlant.
	P1. C'est comme annoter les changements en spécifiant les buts de ceux ayant fait évoluée l'ontologie…c'est comme la valeur sémantique qui est accordée à ce changement.
	E. Oui, je comprends, c'est comme si on demandait à l'utilisateur, pour chaque changement, de le justifier, d'écrire un commentaire qui explique pourquoi il l'a faite. C'est certain que ça aide à la compréhension de l'humain, pas de la machine, mais de l'humain.

REFERENCES

Anderson, T., et Whitelock, D. (2004). The Educational Semantic Web: Visioning and racticing the Future of Education. *Journal of Interactive Media in Education, 1*(Special Issue on the Educational Semantic Web).

Antoniou, G., et van Harmelen, F. (2004). Web Ontology Language: OWL. In S. Staab et R. Studer (Eds.), *Handbook on Ontologies* (pp. 67-92): Springer Verlag.

Aroyo, L., et Dicheva, D. (2004). The New Challenges for E-learning: The Educational Semantic Web. *Educational Technology & Society, 7*(4), 59-69.

Arpirez, J., Corcho, O., Fernandez-Lopez, M., et Gomez-Perez, A. (2001). *WebODE : a Workbench for Ontological Engineering*. Paper presented at the First international Conference on Knowledge Capture (K-CAP'01), Victoria, Canada.

Bechhofer, S., et Goble, C. (2001). *Towards Annotation Using DAML+OIL*. Paper presented at the KCAP'01 Workshop on Semantic Markup and Annotation, Victoria, Canada.

Bernaras, A., Laresgoiti, I., et Corera, J. (1996). *Building and Reusing Ontologies for Electrical Network Applications.* Paper presented at the 12th European Conference on Artificial Intelligence (ECAI96).

Berners-Lee, T. (2000). Semantic Web - XML 2000 Conference. from http://www.w3.org/2000/Talks/1206-xml2k-tbl/Overview.html

Berners-Lee, T. (2005). Uniform Resource Identifier (URI): Generic Syntax. From http://www.gbiv.com/protocols/uri/rfc/rfc3986.html

Berners-Lee, T., Hendler, J., et Lasilla, O. (2001). The Semantic Web. *Scientific American, 284 (5)*.

Borst, W. N. (1997). *Construction of Engineering Ontologies for Knowledge Sharing and Reuse*. University of Tweenty, Enschede, Centre for Telematica and Information Technology.

Bourguin, G., et Derycke, A. (2005). Systèmes Interactifs en Co-évolution, Réflexions sur les apports de la Théorie de l'Activité au support des Pratiques Collectives Distribuées. *Revue d'Intéraction Homme-Machine, AFIHM Europia, 6*(1), 1-31.

Chandrasekaran, B., Josephson, J. R., et Benjamins, R. (1999). What are ontologies, and why do we need them ? *IEEE Intelligent systems, 14*(1), 20-26.

Charlet, J., Bachimont, B., et Troncy, R. (2003). Ontologies pour le Web sémantique. In J. Charlet, P. Laublet et C. Reynaud (Eds.), *Web sémantique*: Action_spécifique_32_CNRS/STIC.

Collier, N., Kawazoe, A., Kitamoto, A., Wattarujeekrit, T., Mizuta, T., et Mullen, A. (2004). *Integrating Deep and Shallow Semantic Structures in Open Ontology Forge.* Paper presented at the Special Interest Group on SemanticWeb and Ontology.

Dewey, J. (1922). *L'école et l'enfant*. Neuchâtel: Delachaux et Niestlé.

Diehl, W., Grobe, T., Lopez, H., et Cabral, C. (1999). *Project-based learning: A strategy for teaching and learning* Boston, MA: Center for Youth Development and Education, Corporation for Business, Work, and Learning.

Ding, Y., et Fensel, D. (2001). *Ontology Library Systems: The key for successful Ontology Reuse.* Paper presented at the First Semantic Web Working Symposium (SWWS'1), Stanford, USA.

Ding, Y., Fensel, D., Klein, M., Omelayenko, B., et Schulten, E. (2004). The role of Ontologies in eCommerce. In S. Staab et R. Studer (Eds.), *Handbook on Ontologies* (pp. 593-616): Springer Verlag.

Ehrig, M., Haase, P., van Harmelen, F., Siebes, R., Staab, S., Stuckenschmidt, H., et al. (2003). *The swap data and metadata model for semantics-based peer-to-peer systems.* Paper presented at the MATES-2003, First German Conference on Multiagent Technologies., Erfurt, Germany.

Ericsson, K. A., et Simon, H. A. (1980). Verbal reports as data. *Psychological Review, 87*(3), 215-251.

Euzenat, J. (1995). Building consensual knowledge bases: context and architecture. In N. Mars (Ed.), *Towards very large knowledge bases* (pp. 143-155). Amsterdam: IOS press.

Euzenat, J., et all. (2001). *Research challenges and perspectives of the Semantic Web*: EU-NSF strategic workshop.

Fensel, D., Harmelen, F., Horrocks, I., McGuinness, D., et Patel-Schneider, P. (2001). OIL: An Ontology Infrastructure for the Semantic Web. *IEEE Intelligent Systems, 16*(2), 38-45.

Fensel, D., Horrocks, I., vanHarmelen, F., Decker, S., Erdmann, M., et Klein, M. (2001). OIL in a Nutshell [Electronic Version] from http://www.knowledgeboard.com/library/oil.pdf.

Fernandez, M., Gomez-Perez, A., et Juristo, N. (1997). *Methontology: From Ontological Art Toward Ontological Engineering.* Paper presented at the Spring Symposium Series on Ontological Engineering. AAAI97, Stanford, USA.

Flouris, G., Plexousakis, D., et Antoniou, G. (2006). *Evolving Ontology Evolution.* Paper presented at the SOFSEM 2006: Theory and Practice of Computer Science, Merin, Czech Republic.

Fung, P. (1996). Issues in Project -Based Distance Learning in Computer Science. *Journal of Distance Education/Revue de l'enseignement à distance, 11*(2).

Gediga, G., Hamborg, K., et Düntsch, I. (2002). Evaluation of Software Systems. In A. Kent et J. G. Williams (Eds.), *Encyclopedia of Library and Information Science* (Vol. 72, pp. 166-192).

Gomez-Perez, A. (1999). *Ontological Ingineering: A state of the art*. Madrid: Facultad de Informatica, Universidad Politecnica de Madrid.

Gomez-Perez, A. (2003). Ontology Evaluation. In Staab S. et Studer R. (Eds.), *Handbook on Ontologies* (pp. 251-274): Springer.

Gruber, T. (1993). A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition, 5*(2), 199-220.

Gruber, T. (1995). *Toward Principles for the Design of Ontologies Used for Knowledge Sharing.* Paper presented at the International Workshop on Formal Ontology, Padova, Italy.

Grudin, J. (1992). Utility and Usability: Research Issues and Development Contexts. *Interacting with Computers, 4*(2), 209-217.

Gruninger, M., et Fox, M. S. (1995). *Methodology for the Design and the Evaluation of Ontologies.* Paper presented at the Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI-95, Montréal.

Gruninger, M., et Lee, J. (2002). Ontology Applications and Design. *Communication of the ACM, 45*(2), 39-41.

Guarino, N. (1997). Understanding, Building, and Using Ontologies. *International Journal of Human-Computer Studies*.

Guarino, N., et Giaretta, P. (1995). Ontologies and Knowledge Bases: Towards a Terminological Clarification. In Mars N. J. I. (Ed.), *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing* (pp. 25-32). Amsterdam: IOS Press.

Guarino, N., et Welty, C. (2002). Evaluating ontological decision with Ontoclean. *Communication of the ACM, 45*(2), 61-65.

Haarslev, V., et Moller, R. (2001). *Racer user's guide and reference manual (version1.6)*: Technical report. University of Hamburg, Computer Science Department.

Haase, P., et Sure, Y. (2004). *State-of-the-Art on Ontology Evolution*: Technical report, SEKT informal deliverable 3.1.1.b, Institute AIFB, University of Karlsruhe (2004).

Handschuh, S. (2007). Semantic Annotation of Resources in the Semantic Web In R. Studer, Grimm, S., Abecker, A., (Ed.), *Semantic Web Services: Concepts, Technologies, and Applications* (pp. 135-155). Berlin: Springer.

Handschuh, S., Staab, S., et Maedche, A. (2001). *CREAM — Creating Relational Metadata with a Component-Based, Ontology-Driven Framework.* Paper presented at the First International Conference on Knowledge Capture (K-CAP 2001), Victoria, Canada.

Heflin, J. (2001). *Towards the Semantic Web: Knowledge and representation in a dynamic, distributed environment.* Faculty of the Graduate School of the University of Maryland.

Heflin, J., et Hendler, J. (2000). *Dynamic Ontology on the Web*. Paper presented at the AAAI, 17th National Conference on artificial Intelligence.

Heflin, J., Hendler, J., et Luke, S. (1999). Coping with Changing Ontologies in a Distributed Environment. *Ontology Management.*, 74-79.

Hendler, J. (2001). Agents and the Semantic Web. *IEEE Intelligent systems, March/April 2001*, 30-37.

Horridge. M., Knublauch, H., Rector, A., Stevens, R., et Wroe, C. (2004). A Practical Guide To Building OWL Ontologies Using The Protege-OWL Plugin. from http://www.co-ode.org/resources/tutorials/ProtegeOWLTutorial.pdf

Horrocks, I., Sattler, U., et Tobies, S. (1999). *Practical reasoning for expressive description logics.* Paper presented at the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99).

Hotte, R., et Contamines, J. (2007). Communautés de pratique : auteures et utilisatrices des banques de ressources éducatives. In M. Baron, D. Guin et L. Trouche (Eds.), *Environnements informatisés et ressources numériques pour l'apprentissage. Conception et usages, regards croisés* (pp. 669-682). Paris: Hermès/Lavoisier.

IEEE. (1995). *IEEE Guide for Software Quality Assurance Planning* (No. Std. 730.1-1995). New York (USA). IEEE Computer Society.

Jorgensen, A. H. (1989). Using the thinking-aloud method in system development. In G. Salvendy et M. J. Smith (Eds.), *Desingning and Using Human-Computer Interfaces and Knowledge Based Systemes* (pp. 743-750). Amsterdam: Elsevier.

Kilpatrick, W. H. (1918). The project method. *Teachers College Record, 19*, 319-335.

Kim, D., et Lee, S. (2002). Designing Collaborative Reflection Supporting Tools en e-Project-Based Learning Environments. *Journal of Interactive Learning Research, 13*(4), 375-392.

Klein, M. (2004). *Change Management for Distributed Ontologies.* Vrije Universiteit Amsterdam.

Klein, M., et Fensel, D. (2001). *Ontology versioning for the Semantic Web.* Paper presented at the International Semantic Web Working Symposium (SWWS), USA.

Klein, M., Fensel, D., Kiryakov, A., et Ognyanov, D. (2002). *Ontology versioning and change detection on the web.* Paper presented at the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02), Sigüenza, Spain.

Klein, M., Kiryakov, A., Ognyanov, D., et Fensel, D. (2002). *Finding and characterizing changes in ontologies.* Paper presented at the 21st International Conference on Conceptual Modeling, Tampere, Finland.

Klein, M., et Noy, N. (2003). *A component-based framework for the ontology evolution.* Paper presented at the Workshop on Ontologies and Distributed Systems, IJCAI-2003, Acapulco, Mexico.

Koivunen, M.-R. (2005). Annotea and Semantic Web Supported Collaboration (pp. Invited talk at Workshop on User Aspects of the Semantic Web (UserSWeb)). Heraklion, Greece: European Semantic Web Conference (ESWC 2005)

Koper, R. (2004). Use of the Semantic Web to Solve Some Basic Problems in Education: Increase Flexible, Distributed Lifelong Learning, Decrease Teachers' Workload. *Journal of Interactive Media in Education, 6*(Special Issue on the Educational Semantic Web).

Kozaki, K., Sunagawa, E., Kitamura, Y., et Mizoguchi, R. (2007). *Distributed Construction of Ontologies Using Hozo* Paper presented at the Social and Collaborative Construction of Structured Knowledge (workshop WWW'2007), Banff, Canada.

Kuutti, K., et Kaptelinin, V. (1997). *Cognitive Tools Reconsidered: from Augmentation to Mediation.* Paper presented at the Cognitive Technology , CT'97, Aizu, Japan.

Laublet, P., Reynaud, C., et Charlet, J. (2002). *Sur quelques aspects du Web sémantique.* Paper presented at the Deuxièmes assises nationales du GdR I3, Nancy.

Lebrun, M. (2002). *Des technologies pour enseigner et apprendre* (2 ed.). Paris: De Boeck.

Leontiev, A. (1984). *Activité, Conscience, Personnalité* (2 ed.). Moscou: Éditions du Progrès.

Löwgren, J. (1995). *Perspectives on usability* (No. Technical Report LiTH-IDA-R-95-23): Linköping University.

Maedche, A., Motik, B., et Stojanovic, L. (2003). Managing Multiple and Distributed Ontologies in the Semantic Web. *VLDB Journal - Special Issue on Semantic Web, 12*, 286-302.

McGuinness, D. (2000). *Conceptual Modeling for Distributed Ontology Environments.* Paper presented at the ICCS 2000, Darmstadt, Germany.

Meads, J., et Stubbs, D. (2001). Usability Architects. from http://www.usabilityarchitects.com/

Miles, M., et Huberman, A. (2003). *Analyse des données qualitatives, traduction de la 2e édition américaine*. Bruxelles: De Boeck.

Motta, E., Vargas-Vera, M., Domingue, J., Lanzoni, M., et Ciravegna, F. (2002). *MnM: OntologyDriven Semi-Automatic and Automatic Support for Semantic Mark-up.* Paper presented at the Semantic Authoring, Annotation & Knowledge Markup Workshop at ECAI 2002, Lyon, France.

Moursund, D. (1999). *Project-based learning using information technology*. Eugene, OR: International Society for Technology in Education.

Neches, R., Fikes, R., Finin, T., Gruber, T., Patil, R., Senator, T., et al. (1991). Enabling technology for knowledge sharing. *AI Magazine, 12*(3), 16-36.

Nielsen, J. (1993). *Usability engineering*: Boston, Academic Press.

Nilsson, M., Palmer, M., et Naeve, A. (2002). *Semantic Web Meta-data for e-Learning, Some Architectural Guidelines.* Paper presented at the 11th Worl Wide Web Conference, Hawaii.

Noy, N. (2004). Tools for Mapping and Merging Ontologies. In S. Staab et R. Studer (Eds.), *Handbook on Ontologies*: Springer Verlag.

Noy, N., Chugh, A., Liu, W., et Musen, M. (2006). *A Framework for Ontology Evolution in Collaborative Environments.* Paper presented at the 5th International Semantic Web Conference, Athens, USA.

Noy, N., Fergerson, R., et Musen, M. (2000). *The knowledge model of Protege-2000: Combining interoperability and flexibility.* Paper presented at the 12th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2000), Juan-les-Pins, French Riviera.

Noy, N., et Klein, M. (2003). Ontology evolution: Not the same as schema evolution. *Knowledge and Information Systems, 5*.

Noy, N., Kunnatur, S., Klein, M., et Musen, M. (2003). *Tracking Changes During Ontology Evolution.* Paper presented at the 3rd International Semantic Web Conference (ISWC2004), Hiroshima, Japan.

Noy, N., et Musen, M. (2000). *PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment.* Paper presented at the Seventeenth National Conference on Artificial Intelligence (AAAI-2000), Austin, TX.

Noy, N., et Musen, M. (2002). *PROMPTDIFF: A fixed-point algorithm for comparing ontology versions.* Paper presented at the 18th National Conference on Artificial Intelligence (AAAI-2002), Edmonton, Canada.

Noy, N., et Musen, M. (2003a). Ontology Versioning as an Element of an Ontology-Management Framework. *IEEE Intelligent Systems, in press*.

Noy, N., et Musen, M. (2003b). The PROMPT Suite: Interactive Tools For Ontology Merging And Mapping. *International Journal of Human-Computer Studies, 59*(6), 983-1024.

O'Malley, C. E., Draper, S. W., et Riley, M. S. (1984). Constructive interaction: A method for studying human-computer-human interaction. In B. Shackel (Ed.), *Human-computer interaction - INTERACT'84* (pp. 269-274): Elsevier.

Oberle, D., Volz, R., Motik, B., et Staab, S. (2004). An extensible ontology software environment. In S. Staab et R. Studer (Eds.), *Handbook on Ontologies* (pp. 299-320): Springer Verlag.

Ognyanov, D., et Kiryakov, A. (2002). *Tracking changes in rdf(s) repositories.* Paper presented at the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web.

Oliver, D. E., Shahar, Y., Musen, M., et Shortliffe, E. H. (1999). Representation of change in controlled medical terminologies. *AI in Medicine, 15*(1), 53–76.

OntoWeb. (2002a). *A survey on methodologies for developing, maintaining, evaluating and reengineering ontologies* (No. IST-2000-29243 - Deliverable 1.4).

OntoWeb. (2002b). *A survey on ontology tools* (No. IST-2000-29243-Deliverable 1.3).

Paillé, P., et Mucchielli, A. (2003). *L'analyse qualitative en sciences humaines et sociales*. Paris: Armand Colin.

Paquette, G. (2002). *L'ingénierie pédagogique : pour construire l'apprentissage en réseau*. Québec: Presses de l'Université du Québec.

Paquette, G., et Rosca, I. (2002). Organic Aggregation of Knowledge Object in Educational Systems. *Canadian Journal of Learning and Technology, 28*(3).

Paquette, G., et Rosca, I. (2004). *An Ontology-based Referencing of Actors, Operations and Resources in eLearning Systems.* Paper presented at the SW-EL'04: Semantic Web for E-Learning Workshop, Eindhoven, The Netherlands.

Paquette, G., Rosca, I., Mihaila, S., et Masmoudi, A. (2007). Telos, a service-oriented framework to support learning and knowledge management. In S. Pierre (Ed.), *E-Learning Networked Environments and Architectures: a Knowledge Processing Perspective*: Springer-Verlag.

Pinto, S., Staab, S., et Tempich, C. (2004). *DILIGENT: Towards a fine-grained methodology for Distributed Loosely-controllled and evolvInG Engineering of oNTologies. I.* Paper presented at the 16th European Conference on Artificial Intelligence ECAI-2004, Valencia.

Prie, Y., et Garlatti, S. (2004). Méta-données et annotations dans le Web sémantique. In J. Charlet, P. Laublet et C. Reynaud (Eds.), *Le Web sémantique* (Vol. Hors série de la Revue Information -Interaction - Intelligence (I3), 4(1), Cépaduès, Toulouse, 2004, pp. 45-68).

Quivy, R., et vanCampenhoud, L. (1995). *Manuel de recherche en sciences sociales*: Dunod.

Rabardel, P. (1995). *Les hommes et les technologies*. Paris: A. Colin.

Rogozan, D. (2003). Solution techno-pédagogique aux interactions d'apprentissage en ligne entre groupes de projet. Rimouski: 3ème colloque du CIRTA. Le télé-apprentissage dans la société du savoir.

Rogozan, D. (2004a). *Conception d'une méthodologie pour assurer l'évolution de l'ontologie dans un environnement dynamique et distribué* (No. LICEF06NR01). Montréal: Centre de recherche LICEF.

Rogozan, D. (2004b). Modifying semantic annotations according to the evolution of the ontology. Montréal: First prize to the Annual Conference of the LORNET Research Network (Towards the Educational Semantic Web-04).

Rogozan, D., Hotte, R., et Abdulrab, H. (2002). *Modélisation d'un espace dynamique dédié à la réalisation de projets d'apprentissage à distance.* Paper presented at the TICE Symposium on Technologies de l'Information et de la Communication dans les Enseignements d'ingénieurs et dans l'industrie, France.

Rogozan, D., et Paquette, G. (2003). *Ontologie émergente pour le téléapprentissage par projet.* Paper presented at the Colloque DIVA (Développement, intégration et évaluation des technologies de formation et d'apprentissage), Montreal, Cananda.

Rogozan, D., et Paquette, G. (2005a). *Managing Ontology Changes on the Semantic Web.* Paper presented at the IEEE/WIC/ACM International Conference on Web Intelligence (WI'05), Compiegne, France.

Rogozan, D., et Paquette, G. (2005b). *Semantic Annotation in e-Learning Systems based on Evolving Ontologies.* Paper presented at the SW-EL'05: Applications of Semantic Web Technologies for E-Learning, AIED'05 workshop, Amsterdam, The Netherlands.

Rogozan, D., Paquette, G., et Hotte, R. (2003). *Dynamic Approach for Constructing a Specific Collaboration Space among Distant Project Groups.* Paper presented at the E-Learn Conference, 2003, Phoenix, Arizona.

Rogozan, D., Paquette, G., et Rosca, I. (2004). *Gestion de l'évolution de l'ontologie utilisée comme référentiel sémantique dans un environnement de téléapprentissage.* Paper presented at the TICE International Symposium, Compiegne, France.

Schreiber, G., Wielinga, B., et Breuker, J. (1993). *KADS: A Principled Approach to Knowledge-Based System Development*. London: Academic Press.

Slocum, N. (2006). Focus-groupe. In Fondation Roi Baudouin et viWTA (Eds.), *Méthodes participatives. Un guide pour l'utilisateur*.

Staab, S., Studer, R., Schnurr, H. P., et Sure, Y. (2001). Knowledge Processes and Ontologies. *IEEE Intelligent Systems, January-February*, 26-34.

Stojanovic, L. (2004). *Methods and tools for ontology evolution.* University of Karlsruhe, Germany.

Stojanovic, L., Maedche, A., Motik, B., et Stojanovic, N. (2002). *User-driven Ontology Evolution Management.* Paper presented at the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02), Sigüenza, Spain.

Stojanovic, L., et Motik, B. (2002). *Ontology Evolution within Ontology Editors.* Paper presented at the Knowledge Acquisition, Modeling and Management (EKAW), Siguenza, Spain.

Stojanovic, L., Staab, S., et Studer, R. (2001). *eLearning based on the Semantic Web.* Paper presented at the WebNet2001 - World Conference on the WWW and Internet, Orlando, Florida.

Stojanovic, L., Stojanovic, N., et Handschuh, S. (2002). *Evolution of the Metadata in the Ontology-based Knowledge Management Systems.* Paper presented at the Experience Management 2002, Berlin, Germany.

Stuckenschmidt, H., et Klein, M. (2003a). *Evolution management for interconnected ontologies.* Paper presented at the Workshop on Semantic Integration at ISWC 2003, Sanibel Island, Florida.

Stuckenschmidt, H., et Klein, M. (2003b). *Integrity and change in modular ontologies.* Paper presented at the 18th International Joint Conference on Artificial Intelligence, Acapulco, Mexico.

Studer, R. (2003). *The Semantic Web: Methods, Applications and Future Trends.* Paper presented at the IFIP Conference on commerce, business and government, I3E 2003, Sao Paulo, Brazil.

Stumme, G., Maedche, A.,. (2001). *FCA-Merge: Bottom-Up Merging of Ontologies.* Paper presented at the IJCAI '01 17th International Joint Conference on Artificial Intelligence, Seattle, USA.

Stutt, A., et Motta, E. (2004). Semantic Learning Webs. *Journal of Interactive Media in Education, 10*(Special Issue on the Educational Semantic Web).

Sunagawa, E., Mizoguchi, R., et all. (2003). *An Environment for Distributed Ontology Development Based on Dependency Management.* Paper presented at the International Semantic Web Conference (ISWC), Florida, USA.

Sure, Y., Staab, S., et Studer, R. (2004). On-To-Knowledge Methodology (OTKM). In S. Staab et R. Studer (Eds.), *Handbook on Ontologies* (pp. 117-132): Springer Verlag.

Swartout, B., Ramesh, P., Knight, K., et Russ, T. (1997). *Towards Distributed Use of Large-Scale Ontologies.* Paper presented at the Symposium on Ontological Engineering of AAAI, Stanford, California.

Thomas, J. W. (2000). *A Review of Research on Project-Based Learning*: The Autodesk Foundation.

Tricot, A. (2001). Interpréter les liens entre utilisabilité et utilité des documents électroniques. In M. Mojahid et J. Virbel (Eds.), *Les documents électroniques, méthodes, démarches et techniques cognitives* (ed.): Paris : Europia.

Uschold, M., et Gruninger, M. (1996). Ontologies: Principles, Methos and Applications. *Knowledge Engineering Review, 11*(2).

Uschold, M., et King, M. (1995). *Towards a Methodology for Building Ontologies.* Paper presented at the Workshop on Basic Ontological Issues in Knowledge Sharing.

W3C_Consortium. (2004). RDF/XML Syntax Specification (pp. W3C Recommendation, http://www.w3.org/TR/rdf-syntax-grammar/).

W3C_Recommendation. (2006). Namespaces in XML. from http://www.w3.org/TR/REC-xml-names/

W3C_WebOnt. (2004a). OWL Web Ontology Language Guide and Reference. from http://www.w3.org/TR/2004/REC-owl-features-20040210/

W3C_WebOnt. (2004b). OWL Web Ontology Language Use Cases and Requirements. from http://www.w3.org/TR/2004/REC-owl-test-20040210/

Wache, H., Vogele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., et al. (2001). *Ontology-based integration of information - a survey of existing approaches.* Paper presented at the IJCAI Workshop on Ontologies and Information Sharing.

Web_sémantique. (2001). Action spécifique 32 CNRS / STIC. from http://www.lalic.paris4.sorbonne.fr/stic/

WebOnt. (2004). OWL Web Ontology Language Use Cases and Requirements. from http://www.w3.org/TR/2004/REC-owl-test-20040210/

Xuan, D. N., Bellatreche, L., et Pierra, G. (2006). *Versioning Management Model for Ontology-Based Data Warehouses.* Paper presented at the 8th International Conference on Data Warehousing and Knowledge Discovery (DaWak '06), Krakow, Poland.